£1.00

E L B U G

FOR THE ELECTRON

Vol 1 No 2 November 1983

GAMES

*** ROBOT ATTACK**

*** FOOTBALL KRAZY**

PLUS

*** MUSICAL CHRISTMAS CARD**

*** ARTICLES ON**

*** GRAPHICS**

*** WRITING GAMES**

*** COMPUTER AIDED DESIGN**

PLUS

*** UTILITIES**

*** REVIEWS**

AND MUCH MORE

RETURN OF THE DIAMOND

EDITORIAL

THIS MONTH'S MAGAZINE

This is the Christmas issue of ELBUG and the magazine is filled to overflowing this month with programs and articles. Indeed, we have used part of the back cover to fit everything in. Some of the items have a seasonal flavour, for example, a musical Christmas card for you to type in, plus a game we have called Santa's Parcels. We have also included the first part of an article showing how to write a game playing program using this as an example.

There are another three games, all excellent examples of their kind, to entertain you over Christmas, plus the second part of our series on Electron Graphics. To make the maximum use of your Electron's graphics, we have included a really good Computer Aided Design program called ASTAAD. You can use this to produce quite sophisticated drawings as our illustrations show. You will find plenty more to interest you as you read through the magazine this month.

SOFTWARE CLUB

You will see in the supplement, that you receive with ELBUG, details of the Software Club that we operate for BEEBUG members. Each month, the best of the software that is reviewed in the BEEBUG magazine is offered at discount to BEEBUG members. We expect to offer similar discounts to ELBUG readers as soon as software for the Electron becomes more readily available. Program Power have just launched a large range of Software for the Electron, though they were unable to get any of it to us in time for this issue. Some of their BBC micro software is exceptionally good, and we hope to take a look at their Electron offerings in time for the next issue.

AVAILABILITY OF THE ELECTRON

It appears that many people are having difficulty in buying an Electron. W.H.Smith's, the main high street retailer, and computer shops around the country are selling their Electrons as fast as they arrive, though they will put your name on a waiting list if you ask. Acorn say that there has been difficulty in the supplies of some components, while demand has been higher than expected. Acorn also say that you can order your Electron by mail order directly from Vector Marketing, but according to Vector Marketing you will probably have to wait 6 weeks for your micro to be delivered. The address, if required, is Vector Marketing, Dennington Estate, Wellingborough, Northamptonshire NN8 2RL (tel. 0933-79300).

FUTURE ISSUES

As we like to have at least a short holiday over Christmas ourselves, the next issue of the magazine, which you will receive at the beginning of January will be the January/February issue. The next one after that will be the March issue.

With two issues now complete, we would be pleased to receive any comments from you, the readers. In addition, any contributions, short or long, that you might like to write would also be most welcome. Let's be hearing from you.

Mike Williams.

ELBUG MAGAZINE

GENERAL CONTENTS

HINTS & TIPS

PROGRAMS

3D LETTERING

by A. Weston

Screen presentation is important if you want your programs to really look good and this month we present another short utility that enhances the appearance of text on the screen, particularly where it is used for headings and titles.

This short program produces a 3D effect on lettering used for headings, and can considerably improve the look of the title page of programs in which it is incorporated. In this particular version the text is printed in white with a black shadow against a blue background but you could select different colour combinations by altering line 1020 - see User Guide page 107 for further details.

To use the program for your own purposes, you really only need the procedure PROCdouble. This is defined in lines 1000 onwards. To call the procedure you need one or more lines of the kind appearing on lines 80 to 110. In each case three parameters follow the procedure call. The first, in quotation marks, is the text to be printed. The second two parameters give the x and y graphics coordinates of the printing position on the screen. A little experiment is needed here - but remember that in a graphics mode 0,0 is the bottom left hand corner, and 1279,1023 is the top right hand corner.

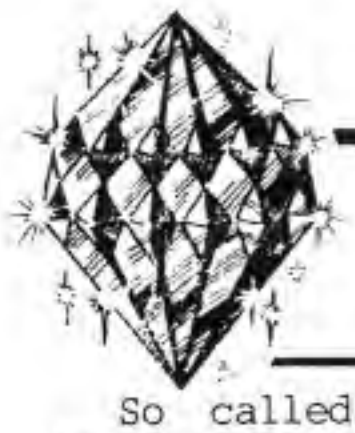
The technique used in this program is to print the text in white, and then to overprint it in black 8 positions down and to the left (hence line 1070). Overprinting might be expected partially to erase the original text, but this is avoided by using the GCOL statement in line 1030 to perform so-called 'OR' plotting. See the User Guide page 153 for more information on this.

```
 10 REM Program  3-D Heading
 20 REM Version  E0.2
 30 REM Author   G.Weston
 40 REM ELBUG    December 1983
 50 REM Program  Subject to copyright
 60:
 70 MODE 5
 80 PROCdouble("ELBUG",450,800)
 90 PROCdouble("for the",400,600)
100 PROCdouble("Acorn",450,400)
110 PROCdouble("Electron",360,200)
120 END
130:
1000 DEF PROCdouble(A$,X,Y)
1010 VDU5
1020 VDU19,0,4,0,0,0:VDU19,1,0,0,0,0
1030 GCOL1,3
1040 MOVE X,Y
1050 PRINT A$
1060 GCOL 1,1
1070 MOVE X-8,Y-8
1080 PRINT A$
1090 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

MOVING GRAPHICS OBJECTS

When moving objects around the screen with MOVE and DRAW remember that a change of several graphics units may be necessary before the shape actually moves one physical graphics dot on the screen. So step the movements by 2, 4 or 8 rather than assuming 1.



RETURN OF THE DIAMOND

by R. McGregor

So called 'adventure games' have become very popular in the last few years. The 'Return of the Diamond' is our very own adventure game now fully adapted for the Electron.

An adventure (in computer terms) is a game where you travel about in a world described to you by the computer. The computer displays where you are, what you can see, and in which directions you can move. You communicate with the computer through a series of English words such as GET, KILL, TAKE etc. Movement, in this particular game can be in any one of four directions, and is accomplished by typing N, S, E or W for the four compass points.

Return of the Diamond is a relatively small adventure game (essential for a game that is to be typed in). There are just nine locations that you can visit. It is nevertheless fun to play, and may give you a feel for adventure games before purchasing a mammoth one. The object of the game is to search for the great diamond, and return it to Diamond Castle. Of course you will meet deadly enemies in your search, and you will need to use your wits to thwart them. As you play the game you will probably need to construct a map of the locations. We will publish the full map next month just in case you get lost.

Happy hunting.

ADVENTURE PROCEDURES

set-up	Sets up the arrays and initialises the variables.
title	Prints the title.
look	Prints the current position, with room description, exits and items.
analyse	Tests for a move, look, inventory, score, quit, or move command.
other-commands	This deals with the rest of the commands.
time-passing	Decreases score, increases moves and deals with the lamp.
move	Moves the player.
inventory	Tells the player of all the items that he is carrying.
take	Takes selected item.
light	Lights the lamp.
off	Turns the lamp off.
drop	Drops the selected item.
kill	Tests if the player has met a gremlin or a pixie.
kill-gremlin	Kills the gremlin.
kill-pixie	Kills the pixie.
finish	End of game.

```
 10 REM Program Diamond
 20 REM Version E0.2
 30 REM Author R.McGregor
 40 REM ELBUG December 1983
 50 REM Program subject to Copyright
 60:
 70 MODE 6:VDU19,0,4,0,0,0
 80 PROCsetup
 90 PROCtitle
100 PROClook
110 REPEAT
120 REPEAT
130 INPUT"What now",c$
140 IF LEN(c$)=0 THEN PRINT"Eh?"
150 UNTIL LEN(c$)>0
160 PRINTSTRING$(39,"-")
170 PROCanalise
180 PROCtimepassing
190 UNTIL dead OR won
200 PROCfinish
210 END
220:
230 DEF PROCsetup
240 DIM place$(9)
250 FOR i=1 TO 9
260 READ place$(i)
270 NEXT i
280 DATA in a hut,in a garden
290 DATA in a shrubbery,on a path
300 DATA on a lane,in a forest
310 DATA at a dead end,at diamond castle
320 DATA in a dark passage
330 DIM newpos(9,4)
340 FOR i=1 TO 9
350 FOR j=1 TO 4
360 READ newpos(i,j)
370 NEXT j
380 NEXT i
390 DATA 0,2,0,0,0,0,5,1,0,0,6,0
400 DATA 0,5,7,0,2,6,0,4,3,0,9,5
410 DATA 4,0,0,0,0,9,0,0,6,0,0,8
420 DIM item$(6),itemname$(6),itempos (6)
430 FOR i=1 TO 6
440 READ item$(i),itemname$(i),itempo s(i)
450 NEXT i
460 DATA a lamp,LAMP,5
470 DATA the great diamond,DIAMOND,7
480 DATA a sharp knife,KNIFE,3
490 DATA a hammer,HAMMER,1
500 DATA a mean looking gremlin,GREML
510 DATA a nasty little pixie,PIXIE,9
520 DIM com$(7)
530 FOR i=1 TO 7
540 READ com$(i)
550 NEXT
560 DATA GET,TAKE,ON,LIGHT,OFF,DROP,KILL
570 DIM direct$(4)
580 FOR i=1 TO 4
590 READ direct$(i)
600 NEXT
610 DATA North,East,South,West
620 DIM bright$(2)
630 bright$(0)="( It's off )"
640 bright$(1)="( It's shining dimly)"
650 bright$(2)="( It's shining bright
ly )"
660 on=FALSE
670 reallit=2.9
680 lit=2
690 position=1
700 dead=FALSE
710 won=FALSE
720 moves=0
730 score=30
740 carried=0
750 ENDPROC
760:
770 DEF PROCtitle
780 PRINT:PRINT
790 PRINT SPC7;"**********************
****"
800 PRINT SPC5;"**********************
********"
810 PRINT SPC3;"****  Return Of The D
iamond  ****"
820 PRINT SPC3;"****     by R. McGreg
or      ****"
830 PRINT SPC5;"**********************
********"
840 PRINT SPC7;"**********************
****"
850 ENDPROC
860:
870 DEF PROClook
880 IF (position=6 OR position=9) AND
 (NOT on OR (itempos(1)<>position AND
itempos(1)<>0)) THEN PRINT"It is pitch
dark.":ENDPROC
890 PRINT
900 PRINT"You are ";place$(position)
910 PRINT
920 PRINT"Exits :"
930 FOR i=1 TO 4
940 IF newpos(position,i)>0 THEN PRIN
T direct$(i);":";
950 NEXT i
960 PRINT
970 PRINT
980 PRINT"You can see :"
990 printed=FALSE
1000 FOR i=1 TO 6
1010 IF itempos(i)=position THEN PRINT
item$(i):printed=TRUE
1020 IF itempos(i)=position AND i=1 AN
D NOT on THEN PRINT bright$(0)
1030 IF itempos(i)=position AND i=1 AN
D on THEN PRINT bright$(lit)
1040 NEXT i
1050 IF NOT printed THEN PRINT"nothing."
1060 ENDPROC
```

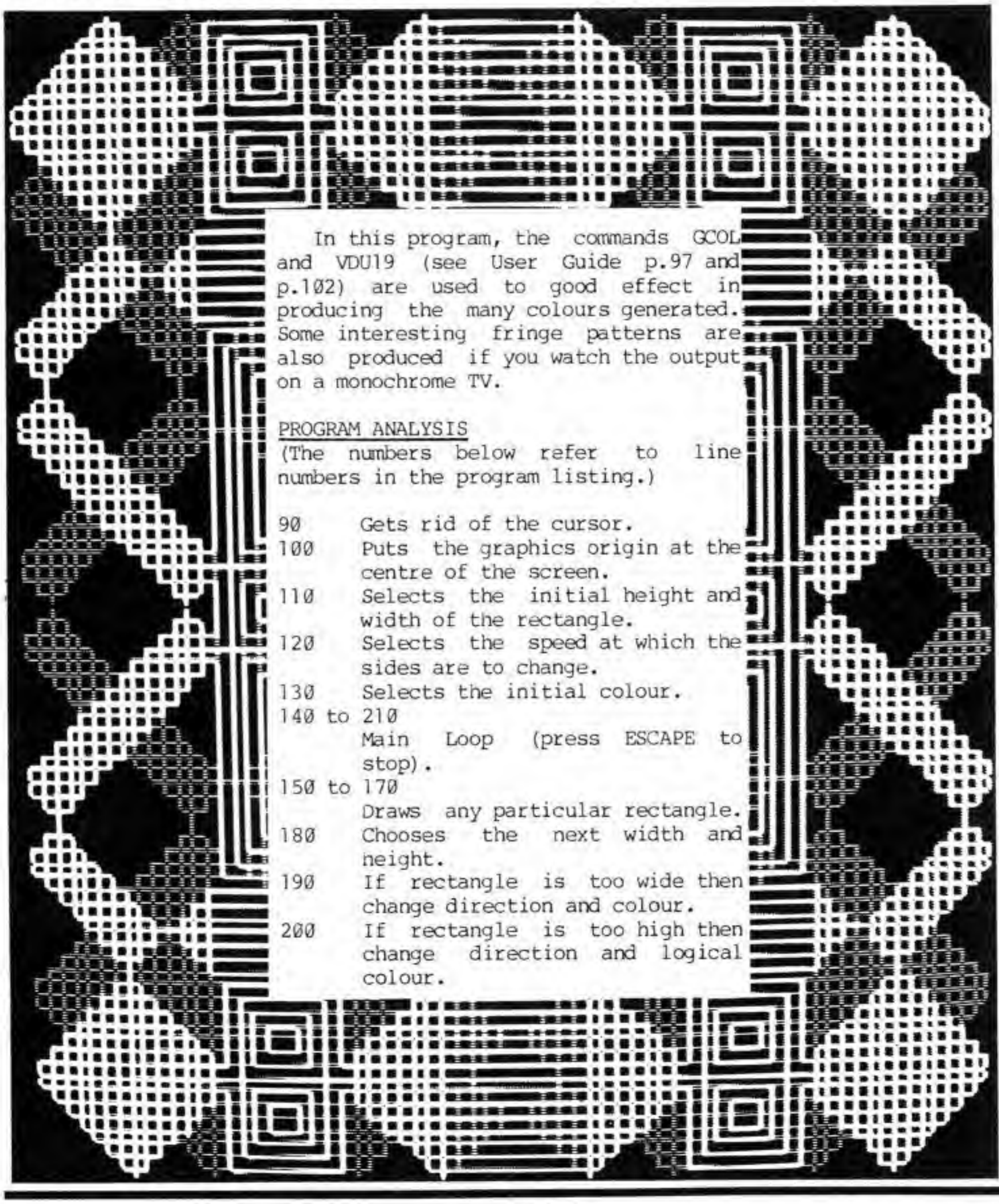
```
 1070:
 1080 DEF PROCanalise
 1090 IF LEN(c$)=1 THEN IF INSTR("NESW"
,c$)>0 THEN PROCmove:ENDPROC
 1100 IF c$="LOOK" THEN PROClook:ENDPRO
C
 1110 IF LEFT$(c$,3)="INV" THEN PROCinv
entory:ENDPROC
 1120 IF c$="SCORE" THEN PRINT"Your sco
re is ";score;".":ENDPROC
 1130 IF c$="MOVES" THEN PRINT"Moves ma
de : ";moves:ENDPROC
 1140 PROCothercommands
 1150 ENDPROC
 1160:
 1170 DEF PROCtimepassing
 1180 score=score-1
 1190 moves=moves+1
 1200 dimmed=FALSE
 1210 IF on THEN reallit=reallit-0.1:di
mmed=TRUE
 1220 lit=INT(reallit)
 1230 IF dimmed AND lit=0 THEN PRINT"Yo
ur lamp just went out.":on=FALSE
 1240 won=(position=8 AND itempos(2)=8)
 1250 ENDPROC
 1260:
 1270 DEF PROCmove
 1280 dir=INSTR("NESW",c$)
 1290 IF newpos(position,dir)=0 THEN PR
INT"You can't move in that direction.":
ENDPROC
 1300 IF (position=6 OR position=9) AND
 (NOT on OR (itempos(1)<>position AND i
tempos(1)<>0)) THEN PRINT"You have fall
en into a snake pit!":dead=TRUE:ENDPROC
 1310 position=newpos(position,dir)
 1320 PROClook
 1330 ENDPROC
 1340:
 1350 DEF PROCinventory
 1360 PRINT
 1370 PRINT
 1380 PRINT"You are carrying :"
 1390 printed=FALSE
 1400 FOR i=1 TO 6
 1410 IF itempos(i)=0 THEN PRINT item$(
i):printed=TRUE
 1420 IF itempos(i)=0 AND i=1 AND NOT o
n THEN PRINT bright$(0)
 1430 IF itempos(i)=0 AND i=1 AND on TH
EN PRINT bright$(lit)
 1440 NEXT i
 1450 IF NOT printed THEN PRINT"nothing
."
 1460 ENDPROC
 1470:
 1480 DEF PROCothercommands
 1490 comno=FNcommand
 1500 thingno=FNthing
 1510 IF comno=0 OR thingno=0 THEN PRIN
T"Sorry.I don't understand.":ENDPROC
 1520 ON comno GOTO 1530,1530,1540,1540
,1550,1560,1570
 1530 PROCtake:ENDPROC
 1540 PROClight:ENDPROC
 1550 PROCoff:ENDPROC
 1560 PROCdrop:ENDPROC
 1570 PROCkill:ENDPROC
 1580 ENDPROC
 1590:
 1600 DEF FNcommand
 1610 no=0:i=0
 1620 REPEAT
 1630 i=i+1
 1640 IF LEFT$(c$,LEN(com$(i)))=com$(i)
THEN no=i
 1650 UNTIL no>0 OR i=7
 1660 =no
 1670:
 1680 DEF FNthing
 1690 no=0:i=0
 1700 REPEAT
 1710 i=i+1
 1720 IF RIGHT$(c$,LEN(itemname$(i)))=i
temname$(i) THEN no=i
 1730 UNTIL no>0 OR i=6
 1740 =no
 1750:
 1760 DEF PROCtake
 1770 IF itempos(thingno)<>position THE
N PRINT"I don't see that here.":ENDPROC
 1780 IF thingno=5 OR thingno=6 THEN PR
INT"You'll be lucky!":ENDPROC
 1790 IF carried=3 THEN PRINT"You can't
 carry any more.":ENDPROC
 1800 itempos(thingno)=0
 1810 PRINT"O.K."
 1820 carried=carried+1
 1830 ENDPROC
 1840:
 1850 DEF PROClight
 1860 IF itempos(thingno)<>0 THEN PRINT
"I would if you had it.":ENDPROC
 1870 IF thingno<>1 THEN PRINT"You're j
oking!":ENDPROC
 1880 IF on THEN PRINT"It's already on.
":ENDPROC
 1890 IF lit=0 THEN PRINT"It won't reli
ght.":ENDPROC
 1900 PRINT"O.K."
 1910 on=TRUE
 1920 ENDPROC
 1930:
 1940 DEF PROCoff
 1950 IF itempos(thingno)<>0 THEN PRINT
"You're not carrying that.":ENDPROC
 1960 IF thingno<>1 THEN PRINT"Come off
 it!":ENDPROC
```

» p17

SQUARE DANCE

by Martin Richards

This program displays a series of expanding and contracting rectangles which change colour during the course of execution. The actual 'form' of the patterns is random and therefore a number of runs will display different patterns. Some patterns repeat after a while, whilst others appear to keep on changing, giving an interesting display of coloured graphics.



In this program, the commands GCOL and VDU19 (see User Guide p.97 and p.102) are used to good effect in producing the many colours generated. Some interesting fringe patterns are also produced if you watch the output on a monochrome TV.

PROGRAM ANALYSIS

(The numbers below refer to line numbers in the program listing.)

90 Gets rid of the cursor.

100 Puts the graphics origin at the centre of the screen.

110 Selects the initial height and width of the rectangle.

120 Selects the speed at which the sides are to change.

130 Selects the initial colour.

140 to 210
Main Loop (press ESCAPE to stop).

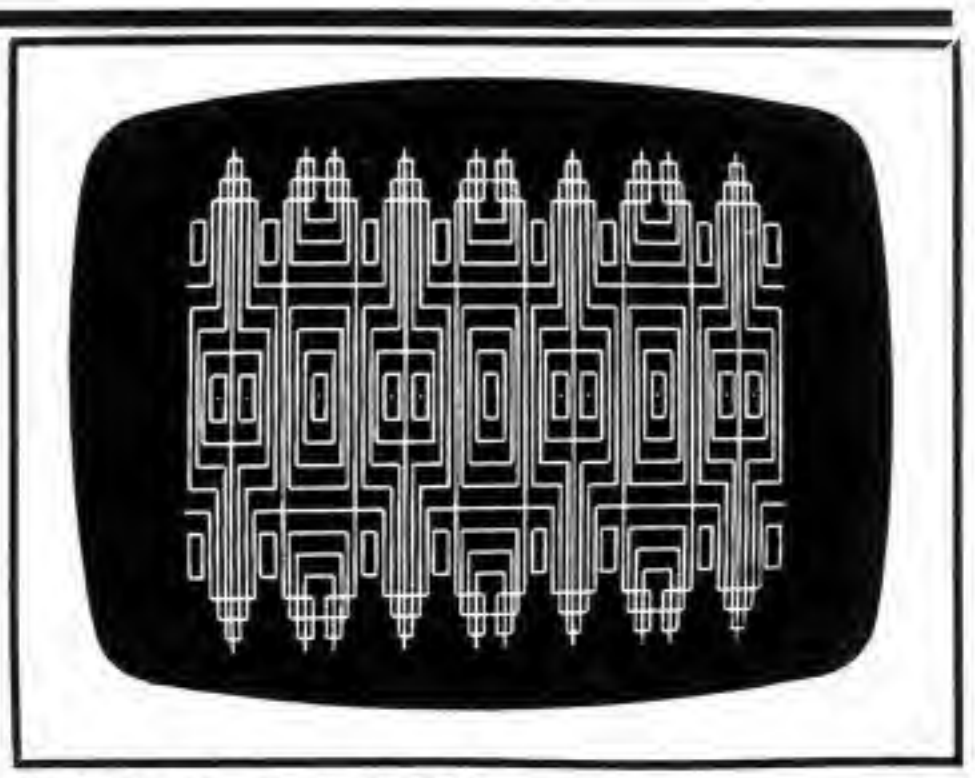
150 to 170
Draws any particular rectangle.

180 Chooses the next width and height.

190 If rectangle is too wide then change direction and colour.

200 If rectangle is too high then change direction and logical colour.

```
 10 REM Program  Square Dance
 20 REM Version  E0.2
 30 REM Author   M.Richards
 40 REM ELBUG    December 1983
 50 REM Program  Subject to copyright
 60:
 70 ON ERROR GOTO 220
 80 MODE 1
 90 VDU23,1,0;0;0;0;
100 VDU29,640;510;
110 X=0:Y=0
120 DX=RND(50) :DY=RND(50)
130 C=RND(3)
140 REPEAT
150 GCOL 3,C
160 MOVE X,Y:DRAW-X,Y:DRAW-X,-Y
170 DRAW X,-Y:DRAW X,Y
180 X=X+DX:Y=Y+DY
190 IF ABS(X)>640 THEN DX=-DX :C=RND(3)
 200 IF ABS(Y)>510 THEN DY=-DY :VDU 19
,RND(3),RND(7);0;
```



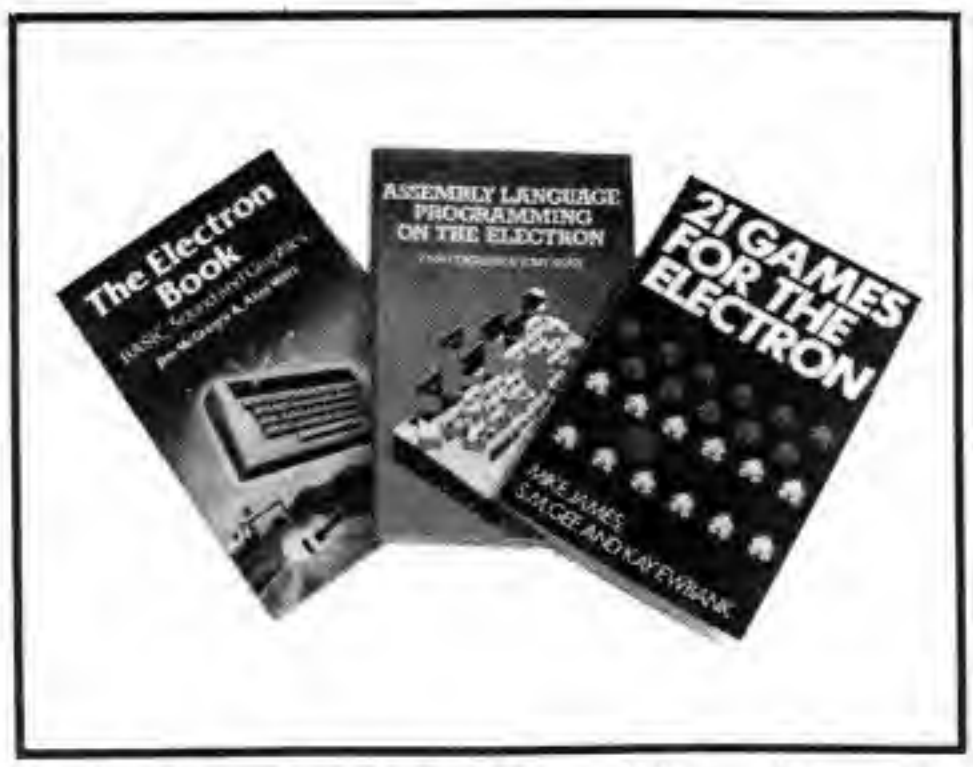
```
210 UNTIL TRUE=FALSE
220 ON ERROR OFF:MODE 6
230 REPORT :PRINT" @ line ";ERL
240 END
```

THREE NEW BOOKS FOR THE ELECTRON

Reviewed by Mike Williams

The arrival of the Electron is the signal for publishers to rush out books for the user, particularly in this pre Cristmas season. We review here, three of the first to be available.

21 Games for the Electron by Mike James, S.M. Gee and Kay Ewbank published by Granada at £5.95.
ISBN 0-246-12344-3.



As well as being a useful collection of games programs this book also emphasises its other aim of developing the programming skills of the reader. In view of this it is really quite astonishing how badly written, in my view, many of the programs are. BBC Basic, which is used on the Electron, is one of the best versions of Basic on any micro for writing clear, readable, well structured programs. Yet this book chooses to ignore nearly all the features which make this such a good Basic, in favour of a style of programming that was out of date 10 years ago. Considerable use is made of subroutines using GOSUB instead of the preferable procedures. REPEAT-UNTIL, so often such a natural form of expression, is never mentioned, neither is IF-THEN-ELSE; there is no use of integer variables, no meaningful variable names, little use of lower case text and many examples of inelegant programming. What is so ironic is that the text accompanying each program is really quite good, with helpful analysis of the program and suggestions for modifications and enhancements. Clearly many Electron users will be tempted by a book of games at the reasonable price of £5.95. Even so I cannot bring myself to recommend a book which makes such bad use of BBC Basic. It is no way to encourage newcomers to computing to write programs.

The Electron Book, Basic, Sound and Graphics by Jim McGregor & Alan Watt published by Addison-Wesley at £7.95.
ISBN 0-201-14514-6

This is an excellent book packed with ideas and information. It is a book for beginners who are interested in writing their own programs and are prepared to spend some time and effort on the way. The first half of the book introduces all the main features of BBC Basic with excellent examples. I would wish that procedures and structured programming were introduced earlier in the book, but it is good to see GOSUB being ignored completely. The second part of the book consists of three chapters dealing with graphics, sound and animation on the Electron. Again there is plenty of detailed information, well presented and with many good illustrations. The style and content of this book are both excellent, and with just over 300 pages for under £8, it is really good value for money. As a comprehensive introduction to programming in Basic on the Electron, this book will be hard to beat.

Assembly Language Programming on the Electron by John Ferguson & Tony Shaw published by Addison-Wesley at £7.95.
ISBN 0-201-14527-8

Although most microcomputers are programmed in Basic, the machine itself can only understand so called 'machine code', a much more primitive language. Programming in Basic is generally much easier, but the necessity of bridging the gap between Basic and machine code, which is what the Electron's Basic interpreter does, tends to make programs written in Basic both long and relatively slow.

Programs that are written directly in machine code are much more compact and much faster. This is how the best of the action games are written, so there are advantages to writing programs in machine code, though it is harder. Such programs are normally written in a slightly easier format called Assembler and this book provides a comprehensive introduction to this form of programming.

Assembler programming is not an easily learnt skill and most books on the subject take a fairly formal approach. The book reviewed here attempts a more popular approach with a number of somewhat gimmicky diagrams and illustrations that unfortunately tend to confuse rather than clarify the ideas being presented. The order of presentation is also somewhat unusual with reference to subroutines and stacks before more than the most elementary addressing modes. Many aspects of assembler programming are covered including, for example, more advanced topics such as interrupts. There is also quite good coverage of the ways in which assembly language programs are embedded within Basic on the Electron.

This book represents an honest and worthwhile attempt to popularise what can often be a difficult subject. Unfortunately, this is not altogether successful in this case, and I continue to believe that a more serious approach is still the best for this subject. That apart, this is a sound book that will help many users to learn assembly language programming and is reasonably priced.

BOOK NEWS

We have also recieved for review a number of books on computing published by Usborne. None of the books is specifically about the Electron but they are all very colourful and well produced and would make ideal Christmas presents for the kids, if not for the grownups. They cost only £1.99 in paperback with a larger than usual page size. Examples of the titles are:

Computer Jargon
Better Basic
Adventure Programs
Robotics
Practise Your Basic

All the books should be in the shops now.

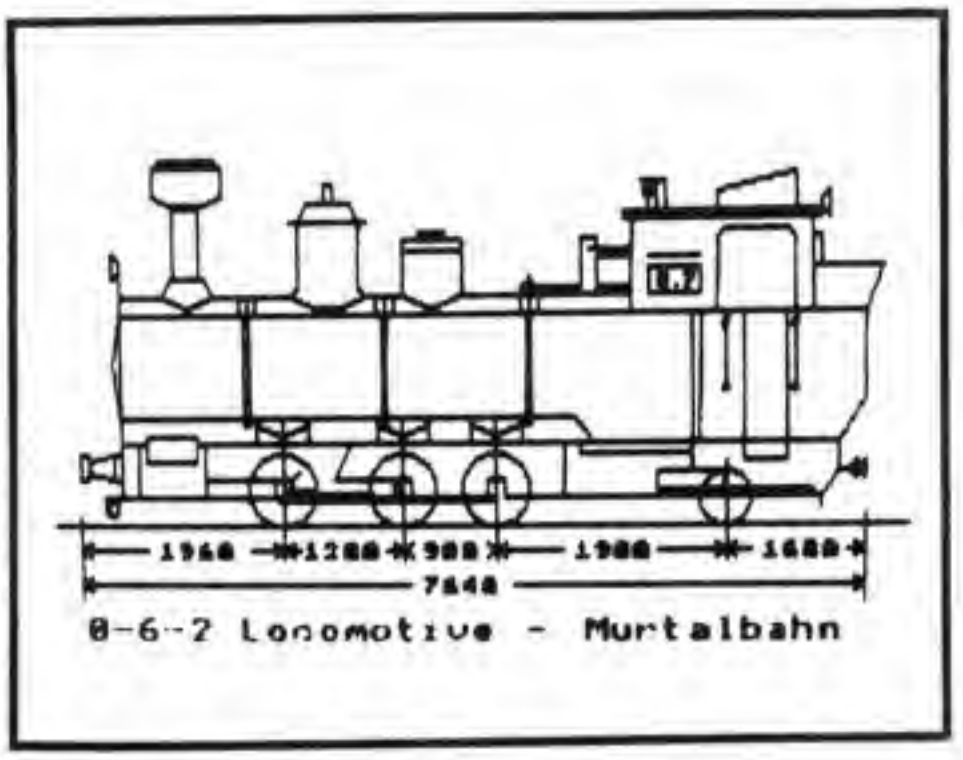
ASTAAD — A PROGRAM FOR COMPUTER AIDED DESIGN

by Jim Tonge

Computer Aided Design (CAD) is one of the more fascinating applications of computers and one for which the Electron has considerable potential. We present here a really excellent example of a CAD package which contains all the basic drawing facilities; not only that, but the package will also allow you to produce Any Size Text printed at any Angle Anywhere on the Drawing, which explains the origins of the name of the package, ASTAAD.

INTRODUCTION

As you can see from the illustrations accompanying this article, the ASTAAD program is capable of producing remarkably sophisticated drawings for its extremely compact size. The program is also well structured so that you can readily extend the range of facilities to suit your own needs. The program runs in mode 0 for the maximum resolution, and all the drawing is done in black on a white background. The various options are selected by the function keys plus Shift and Copy.



0-6-2 Locomotive - Murtalbahn

A function key label accompanies this article. This may be photocopied and attached above the number / function keys with a little Blu Tack or similar.

USING THE PROGRAM

When the program is RUN, a flickering cursor appears at the centre of the screen, which can be moved in steps of 10 g.u.(graphics units) by the four cursor keys. As a guide, the graphics screens on the Electron are made up of 1280 g.u. horizontally, by 1024 vertically. Function key f4 makes the cursor MOVE, f5 makes it DRAW, and f7 DELETES. The distance from the starting point is shown in the top right hand corner, and this acts as a ruler. This can be reset to a new starting point at any time by pressing Shift, and this also happens automatically when you change direction using the cursor keys. The ruler is very useful for measuring parts of a drawing and positioning new objects.

Another method of drawing a line, at an any angle, is to move to one end of the line and press Shift, and then move the cursor to the other end, and press Copy. To delete the line, press Copy again. For greater accuracy, after moving to one end of the line, press f6 and then enter the length in g.u. and angle in degrees, each followed by Return. For example, f6 234 <return> 90 <return> gives a vertical line from the cursor towards the top of the screen; f6 432 <return> 180 <return> gives a horizontal line to the left. Note that Return on its own enters zero, so that f6 432 <return> <return> draws a horizontal line to the right. Also both positive and negative angles can be specified.

The standard characters can be entered directly from the keyboard, the cursor being moved after each one, normally two steps to the right. Underlining needs one step down, superscripts one step up, and the characters ^ / and \ can be positioned for accents.

The "ASTAAD" text facility is brought into use by f0 after moving the cursor to the required position, and 3 inputs are called for. The text input is limited to 41 characters. In response to "Size" any number can be used, but between 2 and 3 is the smallest likely to be legible, depending on the TV/monitor in use,

while 125 almost fills the screen with one character. Any angle can be entered, positive or negative. Note that text can be underlined using the Shift-Copy feature or the f6 routine. Many interesting effects can be discovered by experiment. For example, a row of "size 5 underlines" gives a useful thick line, and with size 8 you can draw a frame around your drawing.

Arrows are often needed on drawings for dimensioning and on leader lines pointing to objects. Function key f1 draws an arrow at the cursor position after the direction angle has been entered.

Function key f2 produces circles, ellipses, triangles, rectangles, pentagons and other polygons to be drawn around the cursor. Three inputs are called for, Horizontal Axis (in g.u.), Vertical Axis, and Number of Sides. A few examples show the way:-

Circle, radius 100 g.u.:
100 <return> 100 <return> 30 <return>
Ellipse:
200 <return> 50 <return> 30 <return>
Square, length of side 300 g.u.:
300 <return> 300 <return> 4 <return>
Regular pentagon:
500 <return> 500 <return> 5 <return>
Vertical line:
<return> 82 <return> 2 <return>

Function key f3 enables you to repeat or copy the last shape (produced using f2) at a new cursor position, and is useful for windows in houses, portholes in ships, shading with dots and angled lines, and so on. If a mistake is made f7 followed by f3 will delete the shape just drawn.

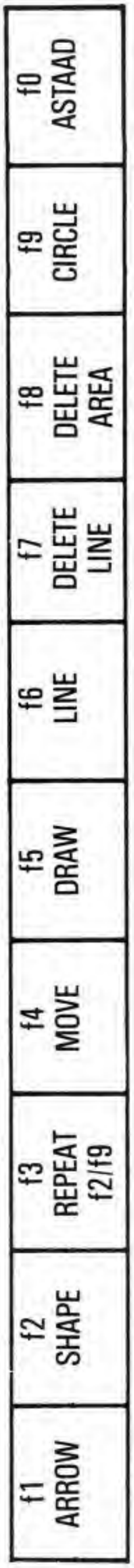
Circles are needed so frequently that f9 is programmed so that only the radius need be entered. Radius=0 (i.e. f9 <return>) prints a dot, and f9 5 <return> produces a useful blob.

To clear an area of the screen, move the cursor to the bottom left hand corner and press Shift, and then move to the top right hand corner and press f8. This will clear the rectangular area defined by these two opposite corners. Reversing the two corners will usually clear the whole screen so be careful! If you want to clear the whole screen then Ctrl-L is the quickest way (i.e. press the CTRL key, and while holding it down, press "L"). Escape can be used if a mistake is made in the middle of a procedure, or to move the cursor to the centre of the drawing area.

APPLICATIONS AND EXTENSIONS

Now is the time to try your own drawing. The possibilities are legion, but here is a tip if you wish to produce a serious drawing rather than a sketch or doodle. Start with a sketch on paper marking dimensions in g.u. Use a scale to suit the available screen area, which is 1250 X 980 g.u., and leave room for a title and labelling.

Other functions and facilities may easily be added to the program. A screen dump to tape is one idea, but above all, frequently-used symbols could be built in. We incorporated a routine into the program to save screen dumps on to tape, in order to provide the illustrations for this article. We also included the facility to load a screen dump back from tape for further use by the ASTAAD program. If you decide to add this feature to your program, then we suggest that you type in and check the main program first.



The easiest way to implement the save/load function is to make the error handling routine call your screen dump (or load) when Escape is pressed, placing the call between lines 1140 and 1160. In this way, pressing Escape and say CTRL will cause a dump to be executed, returning to the screen with the cursor in the central position. In our version Ctrl-Escape produces a 'beep' from the micro. Pressing 'S' will then save the current screen display to tape while pressing 'L' will load a new screen display from tape. In both cases the micro responds with a second 'beep' and the screen display on tape is referred by the standard name 'SCREEN'.

The additional lines that you need for this feature are as follows:

```
1150IF INKEY-2 MOVE -100,-100:VDU7:Ink$
=GET$:VDU7:GOTO1170
1170IF Ink$="L" OSCLI("LOAD SCREEN 3000
") ELSE IF Ink$="S" OSCLI("SAVE SCREEN
3000 7FFF 3000")
1180VDU7:GOTO90
```

Once a screen display has been saved to tape, you can also re-display this directly, without using ASTAAD, by typing:

```
MODE0          <return>
*LOAD SCREEN   <return>
```

You will find that this gives a white image on a black background, the default colours for mode 0. If you want to see the picture displayed as black on white, you must type in the following command after selecting mode and before loading the picture to the screen:

```
VDU19,0,7,0,0,0:VDU19,1,0,0,0,0<return>
```

This reverses the normal use of the two colours black and white in mode 0.

ASTAAD

PROGRAM NOTES

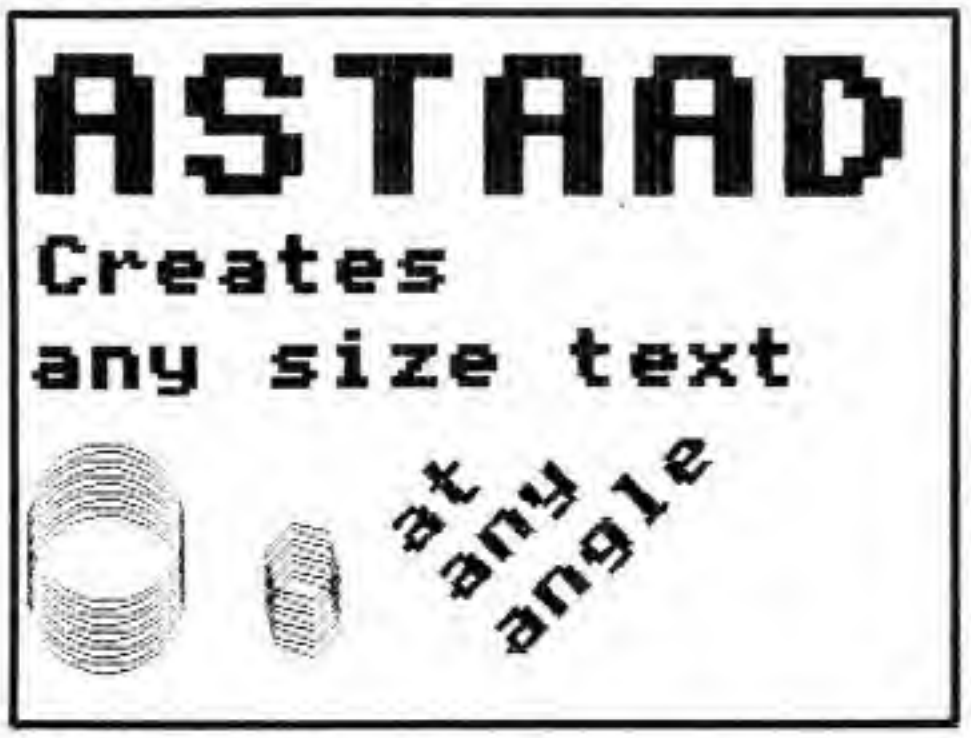
Mode 0 is used in the program for maximum resolution and PROCsetup, called at the start of the program initialises the variables used and positions the cursor at the centre of the screen. *FX4,2 makes the 4 cursor keys and Copy generate ASCII codes, and *FX225,129 causes the function keys to produce ASCII codes 129 upwards. X% and Y% are the cursor co-ordinates with initial mid-screen values.

The main program repeatedly looks for keyboard input while at the same time maintaining the cursor on the screen. Whenever a key is pressed the program distinguishes between normal ASCII codes (less than 128 in value) which are sent direct to the screen (these are mainly printing characters plus the usual control characters), and other inputs such as the function keys, cursor keys, Shift and Copy. For these the program branches to the corresponding procedures and line drawing routines depending on the value of J% (in the range 1 to 16). All cursor keys movements are handled by routines starting at line 250.

PROCtext is the procedure for drawing any size text at any angle. It scans the 8 X 8 matrix for each character in the Electron's ROM (resident Read Only Memory), and moves the graphics cursor to points on the screen depending on the size(S%) and angle(T%) required. Either PROCdot (for small characters less than size 5) or PROCsq (for larger characters) then prints dots or squares to form each character. The procedure is wide open to any input and distortion occurs at some sizes and angles. A user who takes this procedure into his own program is well advised to select a few suitable sizes and amend the spacing (the 7.6 in line 530) and size of square (E and F in line 610).

PROCshape (lines 790-880) is a modified version of 'Fast Polygon' described in BEEBUG Vol.2 No.1, and circles are drawn (f9) by setting up the parameters before calling PROCSHAPE. The other procedures are easy to follow, perhaps with a little aid from the User Guide.

```
  10 REM Program ASTAAD
  20 REM Version E1.3
  30 REM Author Jim Tonge
  40 REM ELBUG December 1983
  50 REM Program subject to copyright
  60 :
  70 MODE 0
  80 ON ERROR GOTO 1140
  90 PROCsetup
 100 REPEAT
 110 PROCruler
 120 IF INKEY-1 THEN key=9:V=5:X1%=X%:
Y1%=Y%:k=10:Z=4:GOTO140
 130 key=INKEY(0):IFkey<=0 THEN 290
 140 IF key<129 PRINT CHR$(key):X%=X%+
20*(X%<1250)*(key<>9):GOTO120
 150 VDU7
```




```
 160 J%=key-128
 170 ON J% GOTO 200,180,220,240,250,25
0,190,250,210,230,120,280,270,270,270,2
70 ELSE 170
 180 PROCarrow:PROCcursor
 190 PROCline:PROCcursor
 200 PROCtext:PROCcursor
 210 PROCdelete:PROCcursor
 220 PROCpoly:PROCsnape(H%,V%,G%):PROC
cursor
 230 PROCcircle:PROCshape(H%,V%,G%):PR
OCcursor
 240 PROCshape(H%,V%,G%):PROCcursor
 250 IF J%>4 AND J%<7 OR J%=8 THEN Z=J
%-1 ELSE Z=4
 260 IF J%=8 Y=7 ELSE Y=5:y=Y-4
 270 X%=X%-10*(J%=13)*(X%>10)+10*(J%=1
4)*(X%<1260):Y%=Y%-10*(J%=15)*(Y%>10)+1
0*(J%=16)*(Y%<990):PLOTZ,X%,Y%
 280 IF J%=12 k=0:PLOTV,X1%,Y1%:IF V=5
 THEN V=7 ELSE V=5
 290 PROCcursor
 300 UNTIL FALSE
 310 END
 320 :
 330 DEFPROCsetup
 340 VDU5
 350 VDU23,1,0;0;0;0
 360 VDU19,0,7,0,0,0:VDU19,7,0,0,0,0
 370 *FX4,2
 380 *FX225,129
 390 X%=630:Y%=490:Y=5:Z=4:J%=0:k=0:B%
=0:T%=0:j=0:X1%=630:Y1%=490
 400 ENDPROC
 410 :
 420 DEFPROCruler
 430 IF J%<13 OR J%>18 ENDPROC
 440 IF k=10 THEN 460 ELSE 450
 450 IF J%<>j THEN X1%=X%:Y1%=Y%
 460 B%=SQR((X1%-X%)^2+(Y1%-Y%)^2)
 470 VDU4:@%=131082:PRINTTAB(60,1)B%+1
0-k:VDU5:j=J%
 480 ENDPROC
 490 :
 500 DEFPROCtext
 510 IF J%<>1 ENDPROC
 520 VDU4,28,6,1,59,1:INPUTTAB(6,1)"Te
xt? "T$,"Size? (2 to 125):  "S%,"Angle(
deg.)?"T%:CLS:VDU5,26
 530 E=S%*SIN(RAD(T%)):F=S%*COS(RAD(T%
))
 540 FOR C%=1 TO LEN(T$)
 550 A%=&BF00 +ASC(MID$(T$,C%,1))*8
 560 FOR P% =0TO 7:FOR Q% =0 TO 7
 570 MOVE X%+P%*E-Q%*F+C%*7.6*F,Y%-Q%*
E-P%*F+C%*7.6*E
 580 IF S%<5 THEN PROCdot ELSE PROCsq
 590 NEXT Q%,P%,C%
 600 ENDPROC
 610 :
 620 DEFPROCdot
 630 IF (2^Q% AND A%?P%)<>0 THEN PLOT6
5,0,0
 640 ENDPROC
 650 :
 660 DEFPROCsq
 670 IF (2^Q% AND A%?P%)<>0 THEN PLOT0
,F,E:PLOT81,(E-F),-(E+F):PLOT81,F,E
 680 ENDPROC
 690 :
 700 DEFPROCarrow
 710 IF J%<>2 THEN ENDPROC
 720 VDU4,28,6,1,59,1:INPUTTAB(6,0)"An
gle of arrow?  "W%:CLS:VDU5
 730 a=15*COS(RAD(W%+35)):b=15*SIN(RAD
 (W%+35)):c=15*COS(RAD(W%-35)):d=15*SIN
(RAD(W%-35))
 740 PLOTy,-a,-b:PLOT0,a,b:PLOTy,-c,-d
:PLOT0,c,d:VDU26
 750 ENDPROC
 760 :
 770 DEFPROCline
 780 IF J%<>7 THEN ENDPROC
 790 VDU4,28,6,1,59,1:INPUTTAB(6,0)"Le
ngth? "L%,"Angle?(deg.) "N%:CLS:VDU5
 800 LX=L%*COS(RAD(N%)):LY=L%*SIN(RAD(
N%)):X2%=X%+LX:Y2%=Y%+LY:PLOTY,X2%,Y2%:
X%=X2%:Y%=Y2%:VDU26
```

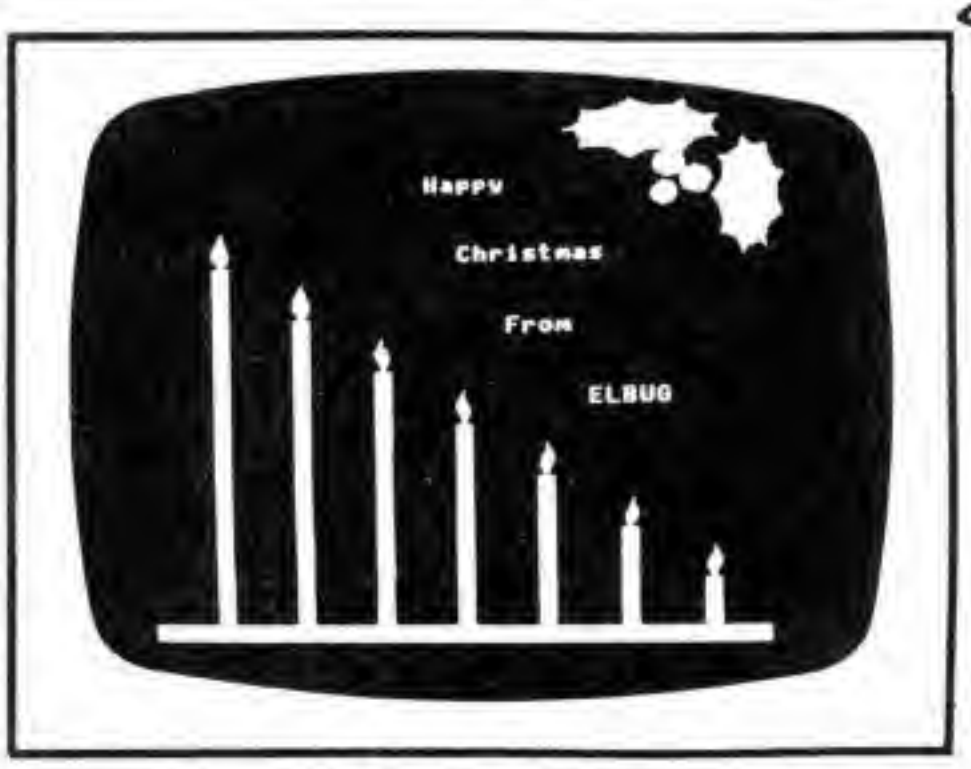
```
  810 ENDPROC
  820 :
  830 DEFPROCpoly
  840 IF J%<>3 THEN ENDPROC
  850 VDU4,28,6,1,59,1:INPUTTAB(6,0)"Ho
rizontal Axis?"H%,"Vertical Axis?"V%,"N
o.of sides(30=circle or ellipse)?"G%:CL
S:VDU5,26
  860 IF G%=4 THEN H%=H%/SQR(2):V%=V%/S
QR(2)
  870 ENDPROC
  880 :
  890 DEFPROCshape(H%,V%,G%)
  900 IF NOT(J%=3 OR J%=4 OR J%=10)THEN
 ENDPROC
  910 LOCAL I%,K%,M%,N,C,S,B,D,R,T
  920 N=2*PI/G%:C=COS(N): S=SIN(N):B=1/
SQR(2):D=1/SQR(2)
  930 FOR I%=1 TO G%+1
  940 R=B*C-D*S:T=B*S+D*C
  950 B=R:D=T:K%=H%*B+X%:M%=V%*D+Y%
  960 IF I%>1 THEN PLOTY,K%,M% ELSE MOV
E K%,M%
  970 NEXTI%
  980 ENDPROC
  990 :
 1000 DEFPROCcircle
 1010 IF J%<>10 THEN ENDPROC
 1020 VDU4,28,6,1,59,1:INPUTTAB(6,0)"Ra
dius of circle? "R%:CLS:VDU5,26
 1030 H%=R%:V%=R%:G%=30
 1040 ENDPROC
```

```
 1050 :
 1060 DEFPROCdelete
 1070 IF J%<>9 ENDPROC
 1080 VDU24,X1%;Y1%;X%;Y%;:CLG:VDU26
 1090 ENDPROC
 1100 :
 1110 DEFPROCcursor
 1120 PLOT4,X%,Y%+15:PLOT6,X%,Y%-15:PLO
T 4,X%-15,Y%:PLOT6,X%+15,Y%:PLOT4,X%,Y%
 +15:PLOT6,X%,Y%-15:PLOT4,X%-15,Y%:PLOT
6 ,X%+15,Y%:PLOT4,X%,Y%
 1130 ENDPROC
 1140 IF ERR<>17 THEN ON ERROR OFF:MODE
7:REPORT:PRINT" at line ";ERL:END
 1160 GOTO 90
```

MUSICAL CHRISTMAS CARD

by D. Chappel

With Christmas approaching we have decided to include a program with a seasonal flavour that provides a good demonstration of the sound and graphics capabilities of your Electron. The program builds up an attractive Christmas card display on the screen and then plays a rendering of "God Rest Ye Merry Gentlemen". We are sure that you will find the results of typing in this relatively short program well worth while. Be careful when copying the program, particularly with the data statements at the end, if you don't want the music to be out of tune.



The program is well structured and you should have no difficulty in identifying the various parts from the procedure names. The holly leaves and berries are drawn using graphics commands while the candles and the flickering flames are drawn with user defined characters. The flickering effect is achieved by randomly switching the flames between several different versions defined at the start of the program. The music is held as a series of numbers in data statements

which are used as the parameters for sound frequency, volume and duration in the procedure PROCmusic. Once started the music will continue indefinitely until you press Escape or Break.

```
   10 REM Program  CANDLES
   20 REM Version  E0.4
   30 REM Author   D.CHAPPEL
   40 REM ELBUG    December 1983
   50 REM Program  subject to Copyright
   60 :
   70 ON ERROR GOTO250
   80 MODE1
   90 PROCsetup
  100 PROCgreeting
  110 PROCholly(1000,950,140,50,0.8)
  120 PROCholly(1200,820,60,100,1.3)
  130 PROCberry(1050,880,24)
  140 PROCberry(1100,850,24)
  150 PROCberry(1040,825,20)
  160 COLOUR1:PROCcandles
  170 COLOUR3:PROCbase
  180 COLOUR2
  190 REPEAT
  200 PROCflicker(RND(7))
  210 PROCmusic
  220 UNTIL Newyear
  230 END
  240 :
  250 ON ERROR OFF
  260 MODE6
  270 IF ERR<>17 REPORT:PRINT" at line
";ERL
  280 END
  290 :
  300 DEF PROCsetup
  310 DIMF$(5),Q%(35),R%(35),S(7),C(7)
  320 PROCchars:Newyear=FALSE
  330 FORI%=1TO5:F$(I%)=CHR$(223+I%*2)+
CHR$(10)+CHR$(8)+CHR$(224+I%*2):NEXT
  340 VDU19,3,2,0,0,0:VDU23,1,0;0;0;0;0
  350 ENDPROC
  360 :
  370 DEF PROCgreeting
  380 COLOUR3:PRINTTAB(18,5)"Happy":COL
OUR1:PRINTTAB(20,9)"Christmas"
  390 COLOUR3:PRINTTAB(23,13)"From":COL
OUR1:PRINTTAB(28,17)"ELBUG"
  400 ENDPROC
  410 :
  420 DEF PROCchars
  430 VDU23,224,&FFFF;&FFFF;&FFFF;&FFFF
F;
  440 VDU23,225,&0;&2020;&3030;&7870;
  450 VDU23,226,&FCF8;&7CFC;&3C7C;&1818;
  460 VDU23,227,&0;&0404;&0C0C;&1E0E;
  470 VDU23,228,&3F1F;&3E3F;&3C3E;&1818;
  480 VDU23,229,&1000;&3010;&7830;&FCFC;
  490 VDU23,230,&7EFE;&3E7E;&1C3C;&1818;
  500 VDU23,231,&800;&C08;&1E0C;&3F3F;
  510 VDU23,232,&7E7F;&7C7E;&383C;&1818;
  520 VDU23,233,&1808;&3C18;&7C3C;&FE7E;
  530 VDU23,234,&FFFF;&FFFF;&3E7F;&181C;
  540 ENDPROC
  550 :
  560 DEF PROCholly(X%,Y%,E%,F%,R)
  570 VDU29,X%;Y%;:GCOL0,3:L%=8
  580 MOVE0,0:MOVE0.9*E%,0
  590 FORA=0TO6.4STEP0.2
  600 MOVE0,0
  610 X%=E%*COS(A)-R*E%/L%*COS(A*L%)
  620 Y%=F%*SIN(A)+1.04/R*F%/L%*SIN(A*L
%)
  630 PLOT85,X%,Y%:NEXT
  640 ENDPROC
  650 :
  660 DEF PROCberry(X%,Y%,B%)
  670 GCOL0,1:VDU29,X%;Y%;
  680 FORA=0TO6.5 STEP0.5
  690 MOVE0,0:MOVEB%*1.2*SINA,B%*COSA
  700 PLOT85,B%*1.2*SIN(A+0.5),B%*COS(A
+0.5)
  710 NEXT:ENDPROC
  720 :
  730 DEF PROCcandles
  740 FORY%=30TO10STEP-1
  750 X%=0:REPEAT:X%=X%+5
  760 PRINTTAB(X%,Y%)CHR$(224)
  770 UNTILX%DIV5=(Y%-7)DIV3:NEXT
  780 ENDPROC
  790 :
  800 DEF PROCbase
  810 FORX%=1TO38:PRINTTAB(X%,31)CHR$(2
24);:NEXT
  820 ENDPROC
  830 :
  840 DEF PROCmusic
  850 IFADVAL(-6)<3 ENDPROC
  860 READP%,D%
  870 IFD%=-1 RESTORE:SOUND1,0,0,30:END
PROC
  880 SOUND1,-10,P%,D%
  890 ENDPROC
  900 :
  910 DEF PROCflicker(K%)
  920 PRINTTAB(K%*5,5+K%*3)F$(RND(5))
  930 ENDPROC
  940 :
  950 REM SOUND DATA
  960 DATA117,7,117,7,145,7,145,7,137,7
,129,7,125,7,117,7
  970 DATA109,7,117,7,125,7,129,7,137,7
,145,21
  980 DATA117,7,117,7,145,7,145,7,137,7
,129,7,125,7,117,7
  990 DATA109,7,117,7,125,7,129,7,137,7
,145,21
```

```
1000 DATA145,7,149,7,137,7,145,7,149,7
,157,7,165,7,145,7
1010 DATA137,7,129,7,117,7,125,7,129,7
,137,7,137,7
1020 DATA129,7,137,7,145,14,149,7,145,
7,145,7,137,7,129,7,125,7,117,14,129,4,
125,4,117,7
1030 DATA137,7,137,7,129,7,137,7,145,7
,149,7,157,7,165,7,145,7,137,7,129,7,12
5,7,117,21,0,-1
```

contd. from page 7

```
1970 IF NOT on THEN PRINT"It's already
off.":ENDPROC
1980 PRINT"O.K."
1990 on=FALSE
2000 ENDPROC
2010:
2020 DEF PROCdrop
2030 IF itempos(thingno)<>0 THEN PRINT
"But you haven't got that.":ENDPROC
2040 itempos(thingno)=position
2050 PRINT"O.K."
2060 carried=carried-1
2070 ENDPROC
2080:
2090 DEF PROCkill
2100 IF itempos(thingno)<>position THE
N PRINT"I don't see that here.":ENDPROC
2110 IF thingno=5 THEN PROCkillgremlin
:ENDPROC
2120 IF thingno=6 THEN PROCkillpixie:E
NDPROC
2130 PRINT"You're joking!"
2140 ENDPROC
2150
2160 DEF PROCkillgremlin
2170 IF itempos(3)=0 THEN PRINT"You sl
ash your knife at the gremlin and kill
it easily.":itempos(5)=-1:score=score+1
0:ENDPROC
2180 IF itempos(4)=0 THEN PRINT"You th
row your hammer at the gremlin,but it c
atches it and throws it back.":ENDPROC
2190 PRINT"You fight the gremlin bare
handed,but only succeed in getting kill
ed."
2200 dead=TRUE
2210 ENDPROC
2220:
2230 DEF PROCkillpixie
2240 IF itempos(4)=0 THEN PRINT"You th
row your hammer at the pixie......A hit
!":itempos(6)=-1:score=score+10:ENDPROC
2250 IF itempos(3)=0 THEN PRINT"You sl
ash your knife at the pixie but it dodg
es.":ENDPROC
2260 PRINT"You fight the pixie bare ha
nded but     it is stronger than you th
ought.You get killed."
2270 dead=TRUE
2280 ENDPROC
2290:
2300 DEF PROCfinish
2310 PRINT
2320 PRINT
2330 IF won THEN PRINT"     Congratula
tions!!!---You won!!!"
2340 IF dead THEN PRINT"     Bad Luck!
!!---You lost!!!":score=0
2350 PRINT
2360 PRINT
2370 PRINT"   You took ";moves;" moves
,"
2380 PRINT"   and your final score was
";score;"."
2390 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SAVING MEMORY

If you often need to use strings in a program, then watch out for your memory, as the Electron will eat up available memory quickly. The reason is that whenever you re-assign a string variable, the Electron compares it to the previous value, and if the new value is longer, it resaves the string again, without clearing the previous value. If, however, the new value is shorter, the old value will be overwritten. Therefore, in the first lines of your programs, before you use any strings that are likely to be reassigned during a program, work out the maximum length a string is likely to be, and define it as a string of spaces slightly larger than required: e.g. suppose the maximum length of A$ is to be 25 characters, then use the following definition: A$=STRING$(26," ").

HOW TO WRITE A COMPUTER GAME (Part 1)

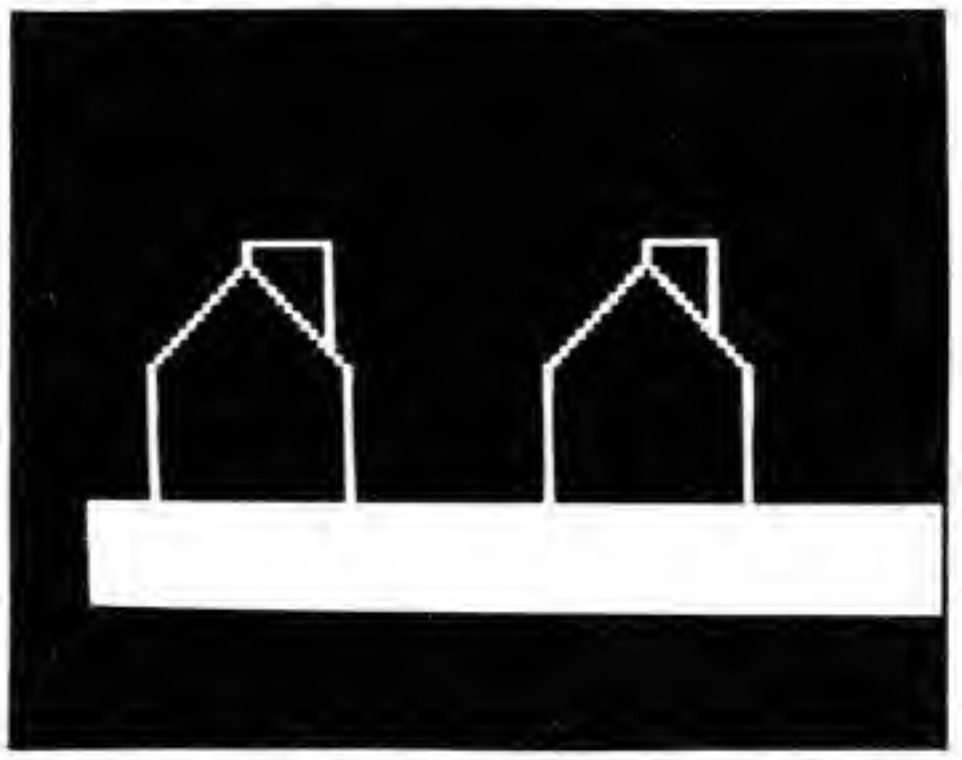
by Mike Williams

Typing in computer games from printed listings can be a useful way to pick up programming ideas that you can use in your own programs. Here we go one better, and show you how to design and write a simple game, explaining each step of the way. The game is called "Santa's Parcels", and a full listing appears elsewhere in this issue.

Clearly not all games programs will be the same, but the task of putting together a games program, indeed almost any program, involves a similar approach. There are, of course, many books that you can read about programming in Basic, and several that are specific to the Electron (see our Book reviews in this issue of ELBUG). You can learn a lot in this way and indeed write good programs as a result. Nevertheless, it can be very instructive to see how someone else designs and develops a computer program. This is what I am going to do here, writing a complete games program, starting from the very beginning, and explaining all the steps along the way. I hope that you will find this both helpful and enjoyable, and that you will discover techniques that you can subsequently incorporate into your own programs.

First of all, I will describe the game itself. Of course, you won't necessarily have a precise idea of how the game will finally appear - new ideas often occur as you are developing the program. The idea of this game, remembering that it is a game for Christmas, is as follows. A number of houses will be drawn across the bottom of the screen, each house having a chimney. The chimneys will be of different sizes. Parcels, which will also be of various sizes, will move across the screen from right to left. Our job as player, will be to hit some key on the keyboard, so that the parcel drops through a chimney into the house below. If we are succesful, then our score will increase, while if the parcel misses the chimney, then our score will be reduced. The game continues until, say, 20 parcels have missed, falling to the ground below.

That is the rough outline of the game, which is sufficient for now. Our aim is to write a clear and readable program that will play this game. One way of writing a good program is to start off with something very simple, probably little more than a series of procedure calls (and these will be virtually non existent at this stage), and then progressively add more and more detail until the program is complete. One significant advantage of this approach is that we can continually test the program as we develop it rather than keep all the testing till the end. You will see better what I mean when we actually get started.



As with most programs, we need to decide what mode we are going to use. This in turn depends on how many colours we want to use, what size text and how much memory is available. The amount of memory is a fairly fixed quantity but modes 3 and under use 20k leaving room only for fairly small programs, while modes 4 and over only use 10k of memory leaving room for much larger programs. We will use mode 2, at least to start with, as this gives us the maximum number of colours to use.

Now some first thoughts about the graphics. We started a series about Electron graphics in the first issue of »

ELBUG and you are strongly recommended to read these articles if you are not yet very familiar with the use of graphics on the Electron. You will also find it helpful to refer to the book by Masoud Yazdani (part of your Electron pack), particularly on the use of MOVE, DRAW and PLOT. We shall use these line drawing instructions to draw the houses because they are relatively large structures, but the parcels will be best as user defined characters. They are relatively small and being single characters we can then move across and down the screen as we wish. We have now decided enough to write a skeleton program as follows:

```
100 MODE 2
110 PROCsetup
120 PROCfrontpage
130 PROCchars
140 PROChouses
150:
160 REPEAT
170 PROCplay
180 UNTIL over%
190 END
```

As I said earlier, the program consists almost entirely of calls to procedures that have yet to be written. Indeed we probably have only a vague idea at the moment of what they should contain anyway, but that doesn't matter. Selecting the mode for the program cannot be put inside a procedure and is usually one of the first statements in a program. PROCfrontpage will display some kind of title page and details of how to play the game. PROCchars will define all the user defined characters that we shall use, while PROCsetup will do most of the other humdrum things, like setting initial values for any variables, that need to be done at the beginning of most programs. The procedure PROChouses, will draw the houses across the bottom of the screen ready for the start of the game.

The line with the colon (:) is there simply to provide visual separation between the two parts of the program and is a useful trick for producing a more readable program.

The game itself then consists of repeatedly going through a series of steps, which I have bundled together at the moment as PROCplay (mostly because I'm not sure what the steps will be in detail yet), until the variable 'over%' becomes true. We assume that over% is initially false and that it is set to true when our 20 misses are up. The game then ends, though we might decide later to add an option to start a new game, keeping a record of highest scores.

It is useful, even at this stage, to start defining the various procedures, though the definitions as you will see do very little. We will also include a line in PROCsetup to initialise 'over%'. Here are the procedures:

```
1000 DEF PROCsetup
1010 over%=FALSE
1090 ENDPROC
1099:
1100 DEF PROCfrontpage
1190 ENDPROC
1199:
1200 DEF PROCchars
1290 ENDPROC
1299:
1300 DEF PROChouses
1390 ENDPROC
1399:
1400 DEF PROCplay
1490 ENDPROC
```

I have started numbering the procedures from line 1000 and I have left gaps for the insertion of more instructions later.

As I said, this isn't yet very exciting as most of these procedures do nothing at all. However, if you have typed everything into your Electron and you now run the program, no error messages will be produced (that is if you haven't made any typing mistakes). The screen will clear, because of the MODE2 instruction at the start of the program, and then nothing further will be seen. The program is repeatedly calling PROCplay which also does nothing as yet.

Now this may all seem rather pointless at the moment. Here we are putting all this effort into typing a program into the micro and when we run it nothing happens; and nothing goes wrong with it either! Knowing that this »

framework contains no programming errors, we can now build up the various procedures one by one, testing each one by itself as we do so. Let's start with the front page. We'll just make this display a title on the screen for now. Add the following lines to your program:

```
1110 COLOUR 1
1120 PRINT TAB(6,2)"SANTA'S"
1130 PRINT TAB(6,6)"PARCELS"
1140 COLOUR 2
1150 PRINT TAB(3,20)"Press any key."
1160 G%=GET:CLS
```

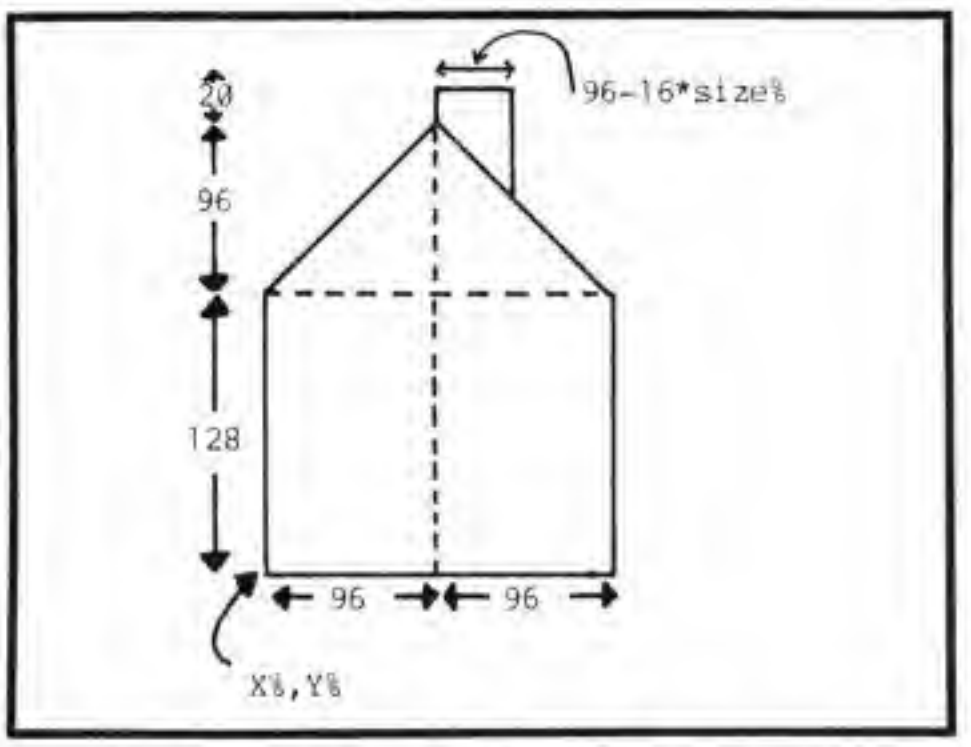
We can immediately test whether this works correctly by simply running the program. It should now display the text in the above instructions on the screen. The TAB statements in lines 1120 and 1130 define whereabouts on the screen the text is to be displayed. At line 1060 the program waits for any key to be pressed, it doesn't matter which, before continuing. If your version doesn't work correctly, then check it very carefully before continuing. The whole point of this approach is that we develop the program in small steps, testing each new addition, so that we always have a correctly working program.

The next step, I think, is to write the instructions to draw the houses. We have already decided that the procedure PROChouses will display all the houses on the screen. However it seems like a good idea to define a further procedure, which we will call PROChouse, to draw just a single house, and then PROChouses can call this to position each house where we want it. This approach has two particular advantages. It breaks the whole task down into two smaller tasks and ensures that the houses will all be drawn exactly the same because they will all be drawn by the same procedure.

To make the procedure PROChouse really useful we will give it four parameters (check the Electron User Guide page 81 if you're not sure about parameters in procedures). The first two will be the position of the house on the screen as graphics co-ordinates (X%,Y%) followed by the size (size%) and the colour (colour%). The size will enable each house to have a different size chimney, as in the description of our game. You will also notice that I am using integer variables (those followed by %) as these use less memory and result in faster programs (see the User Guide page 47 for more information on this). The procedure PROChouse is defined as follows:

```
1500 DEF PROChouse(X%,Y%,size%,colour%)
1510 GCOL0,colour%
1520 MOVE X%,Y%:PLOT 1,0,128
1530 PLOT 1,96,96:PLOT 1,0,20
1540 PLOT 1,96-16*size%,0
1550 PLOT 1,0,16*size%-116
1560 MOVE X%+96,Y%+224
1570 PLOT 1,96,-96:PLOT 1,0,-128
1580 ENDPROC
```

To help you understand these instructions I have drawn out the house with the dimensions below. In mode 2 each character has a width equvalent to 64, and a height equivalent to 32, in graphics terms. Because we shall be defining our own characters for use in this program, I have kept the dimensions of the house mostly in multiples of 32 to assist in lining up parcels with chimneys later on. This didn't all happen just like that, but was the result of some experimenting both on paper and on the screen.



The starting point is the bottom left hand corner so we set the colour for the house using the GCOL command (line 1510) and then move to this position. The simplest way of drawing lines on the screen will often seem to be the DRAW command, but it is worth looking at some of the alternatives

using PLOT instead. If we use PLOT 1,x,y then x and y represent the distance to draw, horizontally and vertically, from the present position, rather than the new co-ordinates. Using this instruction means that what we write for x and y in the program is simpler than would have been the case if we had used DRAW. For example, the PLOT instruction in line 1520 means draw a line such that we move 0 units horizontally and 128 units vertically. To achieve the same result using DRAW would require us to write:

```
DRAW X%,Y%+128
```

Although the difference is small, overall I believe it to be worthwhile.

The other point requiring some explanation is the chimney size. I decided that chimneys would be of size 1, 2, 3, 4 etc corresponding to widths of 80, 64, 48, 32 in graphics dimensions and I then worked out a formula to do this as follows. If the variable is size% (as in this case) then the formula has to be a*size%+b where 'a' and 'b' are two numbers that we now have to find. To do this we use the fact that when size% is 1 the answer should be 80, and that when size% is 2 the answer should be 64 so for

```
size%=1          a + b = 80
size%=2        2*a + b = 64
```

From this we can quickly see that 'a' must be -16 and hence 'b' must be 96. In line 1550 we move downwards and by 20 more than the width of the chimney giving the revised formula used there. This kind of technique occurs quite often in programs where we want one series of numbers to produce a rather different but related series.

We haven't fully linked this procedure into our program yet, but we can test it in immediate mode. After enetering the new procedure, just type:

```
MODE 2 <return>
PROChouse(100,100,1,2) <return>
```

and you should see a green house displayed on the screen. By using different numbers you should be able to display a house of any colour, anywhere on the screen, and vary the size of the chimney as well. Immediate mode can be quite a useful way of testing a procedure in isolation from the rest of the program, but if you try this with your own programs be careful, because it probably won't work if the procedure refers to anything else in your program. Here there is no such problem.

Now that we have a procedure to draw a house, we can complete the writing of the procedure PROChouses to draw all three houses on the screen. The additional lines for our program are as follows:

```
1310 FOR colour%=1 TO 3
1320 PROChouse(384*colour%-320,100,colo
ur%,colour%)
1330 NEXT colour%
1340 GCOL0,7:MOVE 0,0:MOVE 0,100
1350 PLOT85,1280,0:PLOT85,1280,100
```

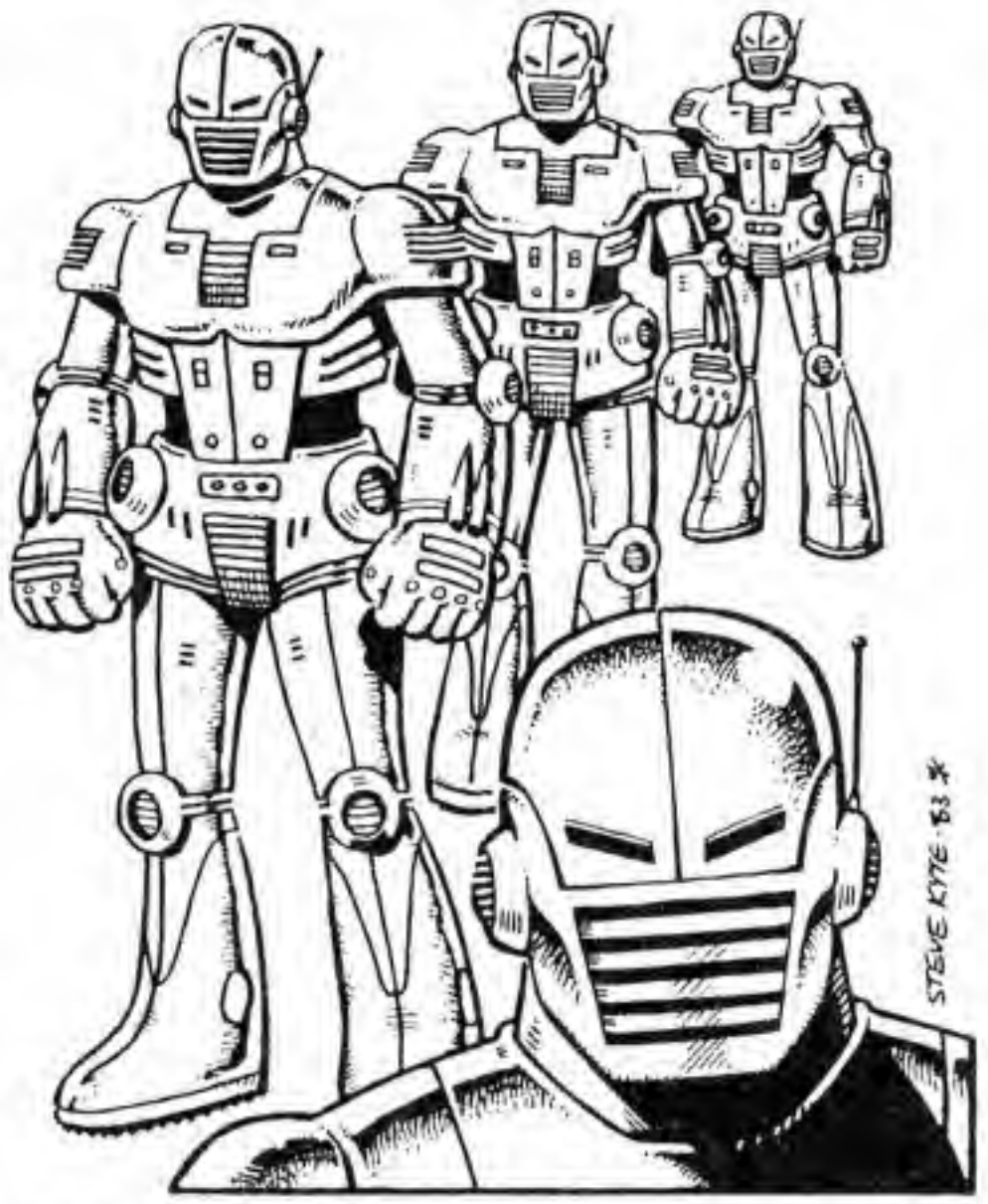
The first three lines above use a FOR-NEXT loop to draw three houses across the screen in red, green and yellow and the last two lines use the triangle fill command (PLOT85,x,y) to paint a layer of white 'snow' underneath the houses. Notice how the value of colour% is used to control the position, colour and size of chimney for each of these houses. You should now be able to run the program, and after the title has been displayed, pressing any key should result in the three houses being displayed. If this doesn't happen then do check the procedures PROChouse and PROChouses very carefully.

Unfortunately, that's all we have space for this month. I have tried to describe everything I have done with this program at some length, and as a consequence the text of this article takes up far more space than even the full program. If you feel I have made everything too simple, remember that there are many Electron owners who are just starting. Next month we will complete our game by adding the moving parcels and scoring. We might even manage to add a few interesting sounds as well. For those of you who just can't wait, we have published the entire program elsewhere in this issue of the magazine.

ROBOT ATTACK

by Edward Hung

This month we present an excellent arcade style game for your entertainment. Like many games, this is addictive with its fascinating mixture of speed and guile.

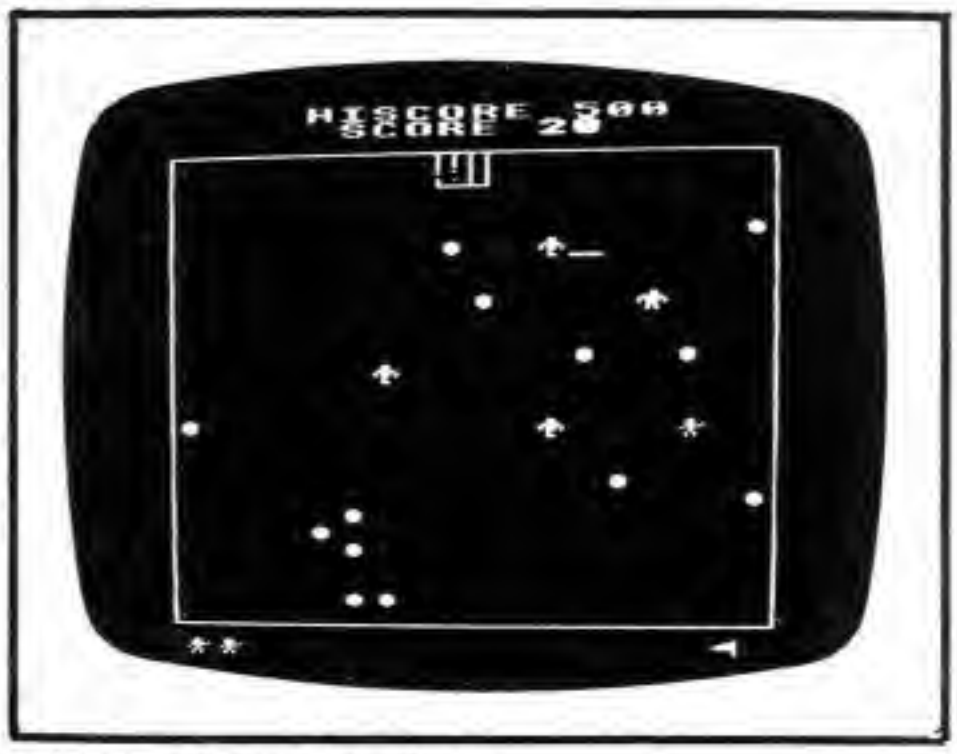


This program is an implementation of the classic zombies game, which has been written for many computers. You control a man who is chased by a number of robots. You must dodge the robots, and at the same time lure them into variously placed mines which explode destroying the robot. You must also keep well clear of the swift and purposeful Marvin. From time to time he will burst through the bars of his cage to pursue you - but you do get a warning - the cage bars and screen-surround will flash if this is imminent. If two robots collide then they are transported through hyperspace to re-appear elsewhere. ROBOT ATTACK is pleasantly fast and fun to play, requiring a nice balance of dextrous skill and strategy. The game has a number of nice optional extras - a sound on/off option, a high score table, instructions and so on.

The program is entirely in Basic. The single CALL command is for OSWORD call zero to input the player's name. This technique is extremely useful as INPUT is rather unsatisfactory for this purpose, allowing any character to be typed. It is also much shorter than trying to write a Basic routine to input a line in this way. The relevant lines are 2730 - 2750 which produce a string N$. The control block is set up to allow a maximum of 20 characters, each in the range ASCII 32 to 128.

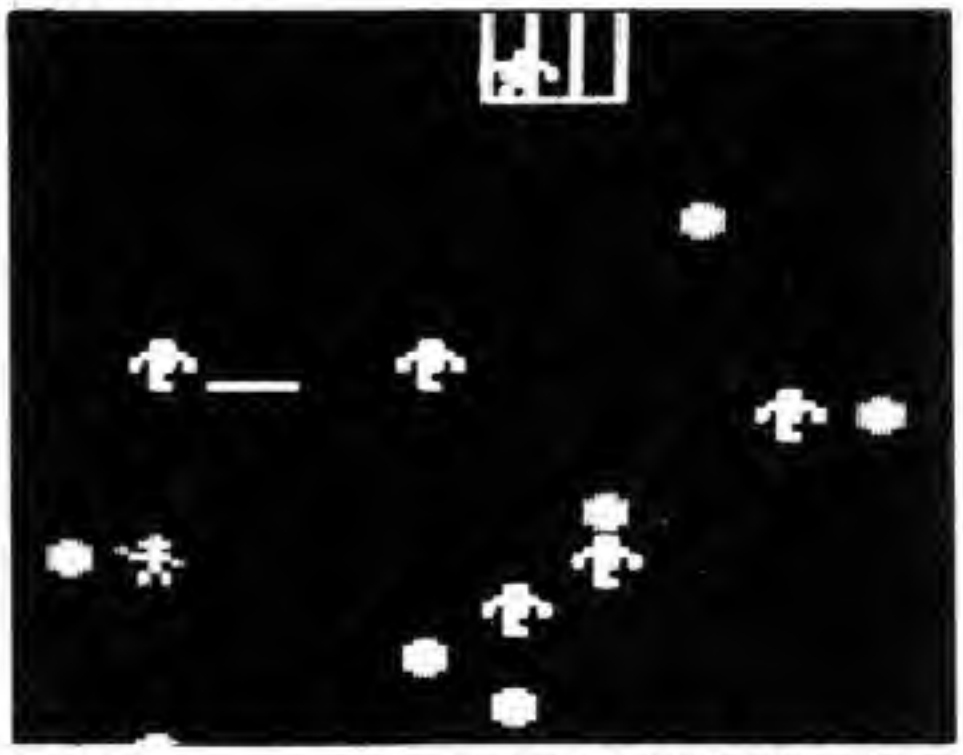

```
 10 REM Program  Robot Attack
 20 REM Version  E0.1
 30 REM Author   E.Hung
 40 REM ELBUG    December 1983
 50 REM Program  Subject to copyright
 60 :
100 PROCsetup
110 ON ERROR GOTO 2990
120 MODE6:PROCtitle
130 REPEAT
140 MODE5
150 SP%=SP%-50
160 PROCobstacles
170 REPEAT
180 I%=I%+1
190 PROCcowboy
200 PROCrobot
210 IF GKL%=1 AND GA%=1 AND DD%=0 AND
RND(1)>.3 PROCmarvin
220 IF I%>4 I%=0
```

```
  230 IF S%>=OS% OS%=OS%+1500:LV%=LV%+1
:PROClv:IF SND%=1 SOUND1,1,100,1
  240 UNTIL DP%=5 OR DD%=1
  250 IF DD%=0 SH%=SH%+1:DP%=0:GOTO150
  260 LV%=LV%-1:DD%=0:IF LV%>0 GOTO160
  270 PROCdelay(3500)
  280 PROCscore:PROCreset:PROCdisplaysc
ore
  290 UNTIL FALSE
  300 END
  310 :
 1000 DEF PROCtitle
 1010 VDU31,12,8:PRINT"ROBOT ATTACK"
 1020 PRINT'TAB(11);"By  Edward Hung"''
"Do you wish to be briefed on the missi
on"TAB(12);"PRESS Y OR N"
 1030 A$=GET$:IF A$="Y" PROCrules ELSE
IF A$<>"N" THEN1030
 1040 ENDPROC
 1050 :
 1060 DEF PROCrules
 1070 VDU12,31,5,1:PRINT"ROBOT ATTACK I
NTRUCTIONS"
 1080 PRINT'"     You are the last surv
ivor of the    Human  Race.  You are  b
eing  pursued"
 1090 FOR T%=1TO13:READ A,A$:PRINTTAB(A
)A$:NEXT
 1100 PRINT'TAB(8)"SPACE BAR TO HYPERSP
ACE"'''TAB(10);"ANY KEY TO START"
 1110 DATA0,"by a hostile group of Bolo
 Bolo Robots,",1,"whose only orders are
 to kill you by",13,"ramming you.",2,"O
 ccasionally, Mean MARVIN, King of",4,"
 Boloans appears to help them.",2,"You m
 ust kill them by luring them"
  1120 DATA2,"onto mines which you yours
 elf must",13,"not walk onto.",3,"CONTRO
 LS:",16,"Z - LEFT",16,"X - RIGHT",16,":
  - UP",16,"/ - DOWN"
  1130 *FX15,0
  1140 A=GET
  1150 ENDPROC
  1160 :
  1170 DEF PROCsetup
  1180 ENVELOPE1,1,0,0,0,0,0,0,126,0,0,-
126,126,126
  1190 VDU23,250,8,28,9,30,40,8,20,20
  1200 VDU23,251,16,56,144,120,20,16,40,
40
  1210 VDU23,252,24,24,60,90,90,16,16,24
  1220 VDU23,253,24,16,124,186,186,16,40
  40
  1230 VDU23,254;24,60,60,60,60,24;
  1240 VDU23,255;146,84;198;84,146
  1250 VDU23,224,160,84,184,16,8,5,2;
  1260 VDU23,225;0;0,9,81,127;
  1270 VDU23,226,5,42,29,16,32,192,64;
  1280 VDU23,227;0;0,72,69,127;
  1290 VDU23,228,1,1,1,1,1,1,1,1
  1300 VDU23,229,0,6,30,62,30;0;
  1310 DIMX%(5),Y%(5),S%(10),N$(10)
  1320 SND%=1:*FX4,1
  1330 PROCscinit
  1340 PROCreset
  1350 ENDPROC
  1360 :
```




```
 1370 DEF PROCscinit
 1380 FOR I%=1TO10:S%(I%)=550-I%*50:N$(
I%)="The Robots of Zaexon":NEXT
 1390 ENDPROC
 1400 :
 1410 DEF PROClv
 1420 VDU17,3,31,1,31:PRINT SPC(14);TAB
(1,31)STRING$(LV%-1,CHR$250);
 1430 ENDPROC
 1440 :
 1450 DEF PROCreset
 1460 S%=0:SC%=0:OS%=1500:SP%=400:SH%=1
:LV%=3:F%=0
 1470 ENDPROC
 1480 :
 1490 DEF PROCrnd1
 1500 A%=RND(18):B%=RND(25)+4
 1510 ENDPROC
 1520 :
 1530 DEF PROCrnd2
 1540 X%(I%)=RND(18):Y%(I%)=RND(25)+4
 1550 ENDPROC
 1560 :
 1570 DEF PROCrnd3
 1580 X%=RND(18):Y%=RND(25)+4
 1590 ENDPROC
 1600 :
 1610 DEF PROCobstacles
 1620 VDU23,1,0;0;0;0;:VDU28,1,29,18,3:
VDU12:VDU26:VDU19,15,2;0;
 1630 GCOL0,3:MOVE60,60:DRAW1219,60:DRA
W1219,930:DRAW60,930:DRAW60,60:FOR I%=5
76TO704STEP33:MOVEI%,926:DRAWI%,866:NEX
T:DRAW576,866
 1640 FOR T%=1TO15:VDU17,1,31,RND(18),R
ND(24)+5,254:NEXT
 1650 FOR I%=1TO5
```

```
 1660 REPEAT PROCrnd2:UNTIL FNpoint=0
 1670 VDU17,6,31,X%(I%),Y%(I%),252:NEXT
 1680 VDU5,18,0,2,25,4,576;1024-4*32;25
3,4,17,5,31,5,0
 1690 PRINT"HISCORE ";S%(1)TAB(6,1)"SCO
RE ";S%
 1700 I%=0:DD%=0:GKL%=-1:DP%=0:D%=250:G
A%=0:FL%=0:C%=0
 1710 PROClv
 1720 LE%=SH%MOD5
 1730 IF SH%/5=SH%DIV5 LE%=5
 1740 IF LE%=1 VDU4:PRINT TAB(14,31);SP
C(4);
 1750 VDU5,18,0,1,25,4,(18-LE%)*64;1024
-31*32;
 1760 IF SH%>30 SH%=1
 1770 IF SH%<31 A=7:IF SH%<26 A=6:IF SH
%<21 A=5:IF SH%<16 A=2:IF SH%<11 A=4:IF
 SH%<6 A=1
 1780 GCOL0,A:PRINTSTRING$(LE%,CHR$(229
))
 1790 GCOL0,3:MOVE(18-LE%)*64,1024-31*3
2:PRINT TAB(18-LE%,31)STRING$(LE%,CHR$(
228))
 1800 REPEAT PROCrnd1:UNTIL FNpnt(A%,B%
)=0
 1810 a%=A%:b%=B%
 1820 VDU4,17,3,31,A%,B%,250
 1830 FOR T%=1TO10000:NEXT
 1840 VDU17,3,31,A%,B%,250
 1850 ENDPROC
 1860 :
 1870 DEF PROCcowboy
 1880 IF INKEY-98  A%=A%-1:F%=1:D%=251
 1890 IF INKEY-67  A%=A%+1:F%=1:D%=250
 1900 IF INKEY-105 B%=B%+1:F%=1
 1910 IF INKEY-73  B%=B%-1:F%=1
 1920 IF INKEY-99  H%=1:F%=1:REPEAT PRO
Crnd1:UNTIL FNpnt(A%,B%)<>6 AND FNpnt(A
%,B%)<>2:IF SND%=1 THEN SOUND17,2,50,10
 1930 IF F%=0 ENDPROC
 1940 A%=A%+(A%>18)-(A%<1):B%=B%+(B%>29
)-(B%<3)
 1950 IF FNpnt(A%,B%)=1 PROCpit(a%,b%,A
%,B%):DD%=1:ENDPROC
 1960 IF FNpnt(A%,B%)=6 OR FNpnt(A%,B%)
=2 PROCdead:DD%=1:ENDPROC
 1970 IF FNpnt(A%,B%)<>0 A%=a%:B%=b%:EN
DPROC
 1980 PRINT TAB(a%,b%)CHR$32:IF H%=1H%=
0:PROCdelay(700)
 1990 VDU17,3,31,A%,B%,D%
 2000 a%=A%:b%=B%:F%=0
 2010 FOR T%=0TO(SP%/50):NEXT
 2020 ENDPROC
 2030 :
 2040 DEF PROCrobot
 2050 IF DD%=1 ENDPROC
 2060 IF X%(I%)=0 AND Y%(I%)=0 ENDPROC
 2070 OX%=X%(I%):OY%=Y%(I%)
 2080 X%(I%)=X%(I%)+(X%(I%)>A%)-(X%(I%)
<A%)
 2090 Y%(I%)=Y%(I%)+(Y%(I%)>B%)-(Y%(I%)
<B%)
 2100 IF FNpoint=6 OR FNpoint=2 REPEAT
PROCrnd2:UNTIL FNpoint=0:IF SND%=1 SOUN
D1,2,50,10
```



```
 2110 IF FNpoint=1 PROCpit(OX%,OY%,X%(I
%),Y%(I%)):X%(I%)=0:Y%(I%)=0:DP%=DP%+1:
PROCms(10):ENDPROC
 2120 IF FNpoint=3 PROCdead:DD%=1:ENDPR
OC
 2130 IF FNpoint<>0 Y%(I%)=Y%(I%)+2
 2140 FOR T%=0TO(SP%/50):NEXT
 2150 IF SND%=1 SOUND3,-15,10,1
 2160 PRINTTAB(OX%,OY%);SPC(1):VDU17,6,
31,X%(I%),Y%(I%),252
 2170 IF RND(1)>0.95 AND FL%=0:GKL%=-GK
L%:C%=GKL%:IF GKL%=1 X%=9:Y%=5:FL%=1:J%
=0:GA%=0:VDU19,15,14;0;:IF SND%=1 SOUND
1,-15,50,1
 2180 IF C%=-1 VDU5,18,3,2,25,4,9*64;10
24-4*32;253,4:PRINTTAB(X%,Y%)SPC(1):C%=
0:IF SND%=1 SOUND1,-15,200,1
 2190 IF FL%=1 J%=J%+1:IF J%=10 VDU5,18
,3,2,25,4,9*64;1024-4*32;253,4,19,15,2;
0;:FL%=0:GA%=1:IF SND%=1 SOUND1,-15,200
,1
 2200 ENDPROC
 2210 :
 2220 DEF PROCmarvin
 2230 XX%=X%:YY%=Y%
 2240 X%=X%+(X%>A%)-(X%<A%):Y%=Y%+(Y%>B
%)-(Y%<B%)
 2250 IF FNpnt(X%,Y%)=15 Y%=Y%+2
 2260 IF FNpnt(X%,Y%)=1 PROCpit(XX%,YY%
,X%,Y%):X%=0:Y%=0:GKL%=-1:VDU5,18,3,2,2
5,4,576;1024-4*32;253,4:PROCms(100):END
PROC
 2270 IF FNpnt(X%,Y%)=3 PROCdead:DD%=1:
ENDPROC
 2280 IF FNpnt(X%,Y%)=6 REPEATPROCrnd3:
UNTIL FNpnt(X%,Y%)=0:IF SND%=1 SOUND3,2
,50,10
```

```
 2290 IF SND%=1 SOUND2,3,50,1
 2300 PRINTTAB(XX%,YY%);SPC(1):VDU17,2,
31,X%,Y%,253
 2310 ENDPROC
 2320 :
 2330 DEF PROCpit(a,b,A,B)
 2340 PRINTTAB(a,b)" "
 2350 IF SND%=1 SOUND0,1,5,1
 2360 VDU17,3,31,A,B,255:FOR L=1TO500:N
EXT:PRINTTAB(A,B);SPC(1)
 2370 ENDPROC
 2380 :
 2390 DEF PROCdead
 2400 IF D%=250 D%=224 ELSE D%=226
 2410 VDU17,3,31,a%,b%,D%:FOR L=1TO500:
NEXT:VDU31,a%,b%,D%+1
 2420 IF SND%=1 SOUND0,1,5,1
 2430 PROCdelay(700)
 2440 IF SND%=1 FOR Q%=1TO5:FOR L%=255T
O200STEP-1:SOUND17,-15,L%,1:NEXT,
 2450 IF SND%=0 PROCdelay(2000)
 2460 ENDPROC
 2470 :
 2480 DEF PROCdelay(T%)
 2490 FOR T=1TOT%:NEXT
 2500 ENDPROC
 2510 :
 2520 DEF PROCms(S)
 2530 S%=S%+S:VDU17,2,31,12,1:PRINT;S%
 2540 ENDPROC
 2550 :
 2560 DEF PROCscore
 2570 FOR I%=1TO10
 2580 IF S%>=S%(I%) AND SC%=0 SC%=I%
 2590 NEXT
 2600 IF SC%=0 ENDPROC
 2610 PROCentername
 2620 FOR P%=10 TO SC%+1 STEP-1:S%(P%)=
S%(P%-1):N$(P%)=N$(P%-1):NEXT
 2630 N$(SC%)=N$
 2640 S%(SC%)=S%
 2650 ENDPROC
 2660 :
 2670 DEF PROCentername
 2680 VDU12,17,2:PRINTTAB(0,6)"Your sco
re is in the"''TAB(6)"Top Ten"''''
 2690 VDU17,6:PRINT" Please enter your"
''TAB(7)"name:"''
 2700 *FX15,0
 2710 VDU23;10,224;0;0;0;
 2720 VDU31,0,20;17,3
 2730 X%=&80:Y%=&A:A%=0
 2740 !&A80=&A00:?&A82=20:?&A83=32:?&A8
4=128
 2750 CALL&FFF1
 2760 N$=$&A00
 2770 ENDPROC
 2780 :
 2790 DEF PROCdisplayscore
 2800 VDU22,4,23,1;1;0;0;0,19,3,0;0;:PR
INTTAB(15)"Highscores"'':VDU19,1,3;0;17
,1
 2810 FOR I%=1TO10:IF I%=10AA$=""ELSEAA
$=" "
 2820 BB$=".....":IF S%(I%)<999BB$="...
...":IF S%(I%)<99BB$="......."
 2830 PRINTTAB(3)AA$;I%;BB$;S%(I%);"...
";N$(I%)'
 2840 NEXT
 2850 VDU19,0,0;0;17,1:PRINTTAB(8)"C to
 close sound channel"'TAB(8)"O to open
 sound channel"''TAB(11)"Sound Channel"
'TAB(11)"Any key to start"'TAB(7)"Or @
to clear score table"
 2860 COLOUR3:IF SND%=1 PRINTTAB(25,26)
"on. "ELSEPRINTTAB(25,26)"off."
 2870 *FX15,0
 2880 A=GET
 2890 IF A=64 PROCscinit:GOTO2800
 2900 IF A=67 OR A=99 PRINT TAB(25,26)"
off.":SND%=0:GOTO2880
 2910 IF A=79 OR A=111 PRINT TAB(25,26)
"on. ":SND%=1:GOTO2880
 2920 ENDPROC
 2930 :
 2940 DEF FNpoint
 2950 =FNpnt(X%(I%),Y%(I%))
 2960 DEF FNpnt(U%,V%)
 2970 =POINT(U%*64+32,1010-V%*32)
 2980 :
 2990 ON ERROR OFF
 3000 MODE6
 3010 *FX4
 3020 IF ERR=17 END
 3030 REPORT:PRINT" at line ";ERL
 3040 END
```

This program was adapted from a version for the BBC micro first published in BEEBUG Vol.2 No.3.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

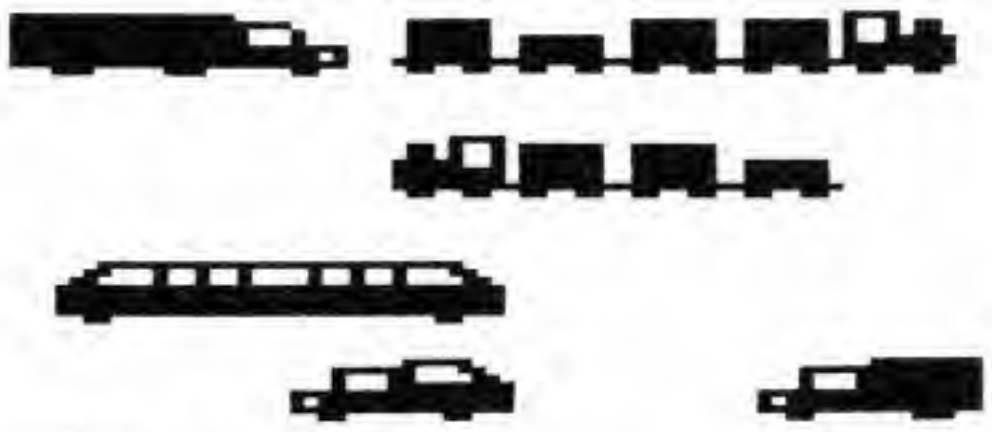
DISABLING THE ESCAPE KEY & DESTROYING PROGRAMS

*FX 200,0 Re-enables the Escape key.
*FX 200,1 Will disable the Escape key.
*FX 200,2 enables the Escape key, but pressing Break will clear the memory, preventing the program being retrieved with OLD.
*FX 200,3 disables the Escape key, and clear the memory on Break.

ELECTRON GRAPHICS (Part 2)

by David Graham

We continue our series on Electron Graphics by introducing multiple user-defined characters, and simple animation techniques.



LARGER CHARACTERS.

Larger characters may be created by joining two or more user defined characters together. The excellent game Hedgehog, published in ELBUG issue 1, uses this principle. A whole series of characters are defined in the procedure PROCDEFINE (line 430 onwards). Many of the characters defined here are small parts of cars and trains which are built up elsewhere in the program.

Take, for example, characters 232 and 233 (defined in lines 620 & 630, see ELBUG issue 1 page 32). These consist of the front and back of a lorry. The lines below will print the two halves separately.

```
100 MODE 5
110 VDU23,232,0,240,136,136,255,253,255
,12
120 VDU23,233,255,255,255,255,255,255,2
55,28
130 PRINT CHR$233
140 PRINT CHR$232
```

You can print the two side by side, and so produce a lorry, by inserting a semicolon (;) at the end of line 130.

You can make the lorry longer by adding a second body piece. (PRINT CHR$233;CHR$233;CHR$232). This technique can be both cumbersome and slow to execute. Fortunately there is a simple solution: use a so-called string variable to hold the string of characters.

Try the following (note that lines 100 to 120 are as above):

```
100 MODE 5
110 VDU23,232,0,240,136,136,255,253,255
,12
120 VDU23,233,255,255,255,255,255,255,2
55,28
130 lorry$ = CHR$233+CHR$233+CHR$232
140 PRINT lorry$
```

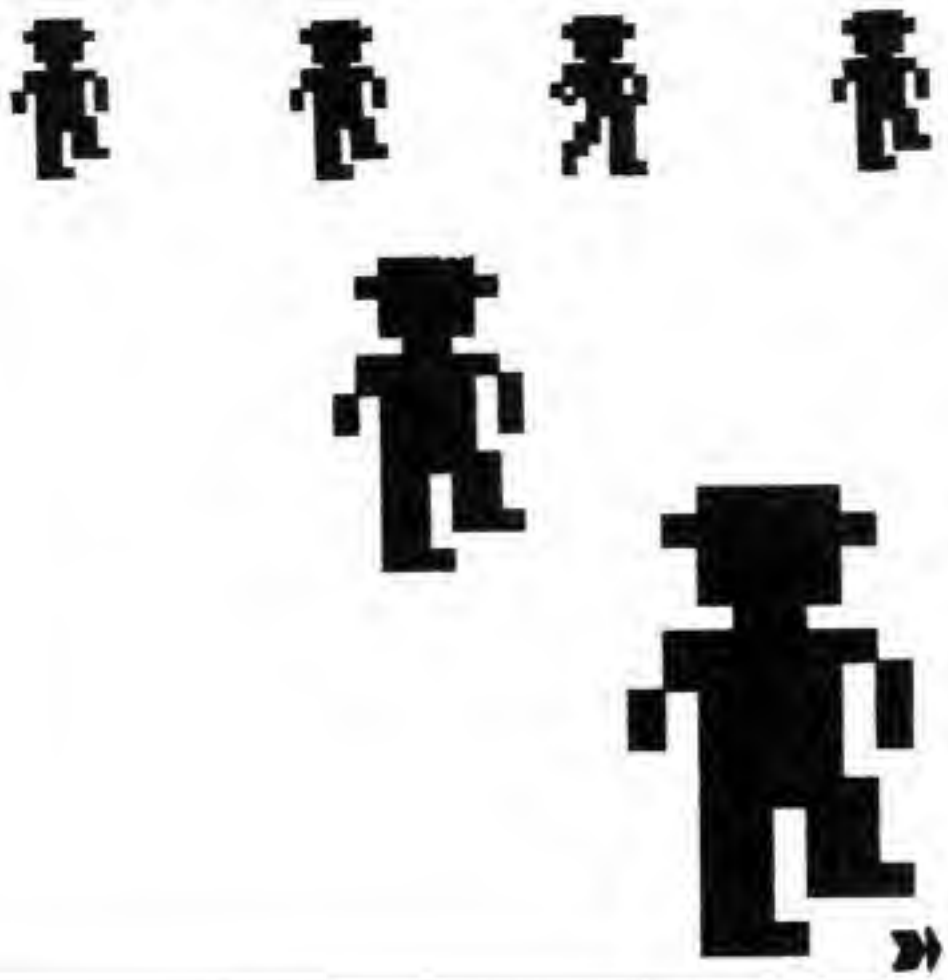
The lorry will now be placed on the screen every time that the expression PRINT lorry$ is encountered. This shows how easy it is to write understandable code in BBC Basic.

If you want to move the lorry across the screen, you must erase the back end with each new position printed, as explained last month. To achieve this you can incorporate a space at the back end of the lorry which will automatically do this for you. Just add the following:

```
150 lorry$ = CHR$32 + lorry$
```

The character for a blank space already exists in the machine as character 32.

To make the lorry move from left to right across the screen, you just need a FOR loop of the kind used to move the invader in last month's article.



Add the following to achieve this:

```
160 FOR A% = 1 TO 36
170 PRINT TAB(A%,10) lorry$
180 TIME = 0 : REPEAT UNTIL TIME =10
190 NEXT
200 END
```

In the program Hedgehog, whole strings of characters are built up using this technique to create a highway full of moving cars, and a busy railway track. If you have a copy of the program, you may find it interesting to examine the characters that the author has used. To do this, run Hedgehog until the game proper begins (to define the characters), and then press Break. Then type in and run the following program :

```
10 MODE 6
20 FOR A% = 224 TO 255
30 PRINT A% ; CHR$A%
40 NEXT
50 END
```

It will list out the characters used against the character numbers.

VERTICAL CHARACTERS

All of the characters in Hedgehog are constructed by adding individual characters together horizontally. If you wish to create figures that are more than one character high, then a further technique is required to join the two pieces together. In practice it is quite easy. You use cursor control characters to reposition the cursor between each character printing.

In order to illustrate this, let us try to create a 'man', two characters high. The first thing to do is to use the character definer program supplied on our introductory cassette to generate a top and bottom half of the man. The two characters can then be incorporated into the program and printed out. The following gives an acceptable image in mode 4. It will need slimming down somewhat for mode 5. Try the program below:

```
100 MODE 4
110 VDU23,224,60,126,60,60,24,126,125,1
89
120 VDU23,225,189,60,62,54,54,55,48,56
130 PRINT TAB (10,10)CHR$224;CHR$225
```

You will see that the two halves are printed side by side. The neatest way to get them printed correctly is to define a string which repositions the cursor in the correct position for printing the second half. Try adding the following:

```
140 move$=CHR$10+CHR$8+CHR$8+CHR$32
150 man1$=CHR$32+CHR$224+move$+CHR$225
160 PRINT TAB(10,10) man1$
```

This will print a complete man. It works because move$ is made up of characters which reposition the cursor for printing the man's lower half. CHR$10 sends the cursor down one place, while each CHR$8 causes a backspace. We have backspaced twice and added the blank character 32 ready for some animation.

There are four cursor movement characters as shown in the table below.

CHARACTER	EFFECT
8	Backspace one space
9	Forward one space
10	Down onespace
11	Up one space

ANIMATION

The basic principle behind character animation is to cycle through a number of different images to produce the impression of movement. In computer games this is often combined with movement of the character across the screen.

To produce a moving animated man we first need a second image of the man. To keep things simple we will just alter the lower half. A suitable »

alternative character may be defined with

```
130 VDU23,226,95,30,22,54,54,102,70,71
```

If you place this into the program and add the following lines you should see the man walk across the screen.

```
160 man2$=CHR$32+CHR$224+move$+CHR$226
170 PRINTTAB(10,10)man1$;TAB(12,10)man2
$
180 FOR A=1 TO 38
190 PRINT TAB(A,15)man1$
200 X=INKEY(10)
210 PRINT TAB(A,15)man2$
220 X=INKEY(10)
230 NEXT
240 END
```

This is achieved by alternating between the two men (lines 190 & 210) within the FOR loop which moves them across the screen.

You should be able to use these ideas as the basis for some interesting animations - but remember, if your man needs also to walk from right to left, he will need two further sets of legs with feet pointing in the other direction. The use of the INKEY statements in lines 200 and 220 is to slow down the movement. If you remove them, the man moves far too quickly, and the effect is lost. The INKEY statement is in fact a very useful way of creating a time delay. The length of the delay is determined by the number in brackets. This gives the delay in hundredths of a second.

Next month we will look at the creation of multi-coloured characters and their animation. Below you will find a complete listing of the moving man program.

```
  10 REM Program MANRUN
  20 REM Version v3
  30 REM Author D.Graham
  40 REM ELBUG December 1983
  50 REM Program subject to copyright
  60 :
 100 MODE4
 110 VDU23,224,60,126,60,60,24,126,125,
189
 120 VDU23,225,189,60,62,54,54,55,48,56
 130 VDU23,226,95,30,22,54,54,102,70,71
 140 move$=CHR$10+CHR$8+CHR$8+CHR$32
 150 man1$=CHR$32+CHR$224+move$+CHR$225
 160 man2$=CHR$32+CHR$224+move$+CHR$226
 170 PRINTTAB(10,10)man1$;TAB(12,10)man
2$
 180 FORA=1TO38
 190 PRINTTAB(A,15)man1$
 200 X=INKEY(10)
 210 PRINTTAB(A,15)man2$
 220 X=INKEY(10)
 230 NEXT
 240 END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

READABLE PROGRAMS

If you have trouble in reading your programs, try using the LISTO option which inserts spaces at certain selected points according to the table below:

LISTO 0 - list the program as stored in memory
LISTO 1 - insert a space after each line number
LISTO 2 - indent FOR..NEXT loops with two spaces
LISTO 4 - indent REPEAT..UNTIL loops

SO, to insert a space after each line number and indent only FOR..NEXT loops, use 1+2, i.e. LISTO 3. Note, however, that the program will not actually be listed until you type LIST<return>.

RECALL OF FUNCTION KEYS - David Moncur

It is sometimes useful to recall the information associated with a function key and make it appear on the screen. This can be difficult if it is a long string that includes CLS. However it can be done neatly using AUTO: Choose a range of line numbers outside your current program, e.g. from 9000 onwards. Type AUTO 9000 <return>. Pressing each function key followed by return causes the list to be displayed as program lines. Remember to delete them afterwards.

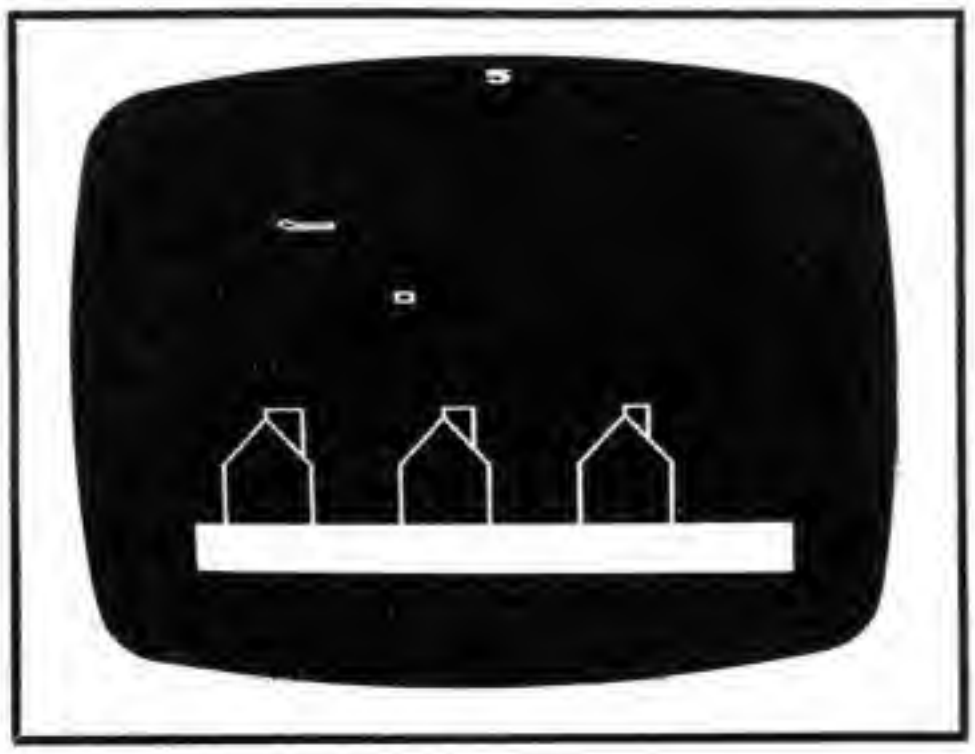
SANTA'S PARCELS

A Game for Christmas by Alan Webster

In this issue of ELBUG we have included the first part of a two part article on how to write a games program. Here, we present the completed program for those who would like to play this game and see the whole program without waiting for next month's issue of the magazine.

The game is quite simple to play. Parcels of different sizes, and placed on a sledge, race across the sky above three houses. By pressing the space bar, you must try to drop a parcel through the chimney of a house below. The parcels are of different sizes and won't necessarily fit all the chimneys. This is recognised in the scoring, as you get 10 points for dropping a parcel through the largest chimney, 25 points for the middle size chimney and 50 points for the smallest chimney. Unfortunately, you also lose 5 points whenever a parcel misses or won't fit the chimney of the house you have chosen. The game continues until 20 parcels have missed their target.

The program has deliberately been kept short because of its role in illustrating the development of a computer game. Other features could easily be added and some of these will be described in the second part of that article in next month's issue of ELBUG.



```
 10 REM Program  SANTA
 20 REM Version  B0.6
 30 REM Author   Alan Webster
 40 REM BEEBUG   December 1983
 50 REM Program  Subject to copyright
 60:
100 MODE2
105 ON ERROR GOTO 500
110 PROCsetup
120 PROCfrontpage
130 PROCchars
140 PROChouses
150:
160 REPEAT
170 PROCplay
180 UNTIL over%
190 PROCend
200 END
210:
500 ON ERROR OFF:MODE 6
510 IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
520 END
530:
1000 DEF PROCsetup
1010 score%=0:miss%=0
1020 drop%=FALSE:over%=FALSE
1030 AX%=19:AY%=RND(10)
1040 parcel=RND(3)+225
1050 VDU23,1,0;0;0;0;
1060 ENVELOPE 1,5,-10,5,20,10,5,20,126
,0,0,-126,126,126
1090 ENDPROC
1099:
1100 DEF PROCfrontpage
1110 COLOUR 1
1120 PRINT TAB(6,2)"SANTA'S"
1130 PRINT TAB(6,6)"PARCELS"
1140 COLOUR 2
1150 PRINT TAB(3,20)"Press any key."
1160 G%=GET:CLS
1190 ENDPROC
1199:
1200 DEF PROCchars
1210 VDU23,224,0,0,0,48,72,71,32,31
1220 VDU23,225,0,0,0,0,0,255,1,255
1230 VDU23,226,248,136,136,136,248,0,0
,0
1240 VDU23,227,0,240,144,144,240,0,0,0
1250 VDU23,228,0,0,224,160,224,0,0,0
1290 ENDPROC
1299:
1300 DEF PROChouses
1310 FOR colour%=1 TO 3
1320 PROChouse(384*colour%-320,100,col
our%,colour%)
```

```
 1330 NEXT colour%
 1340 GCOL0,7:MOVE 0,0:MOVE 0,100
 1350 PLOT85,1280,0:PLOT85,1280,100
 1360 PLOT85,0,0
 1390 ENDPROC
 1399:
 1400 DEFPROCplay
 1410 PROCmove
 1420 IF INKEY-99 AND NOT drop% THEN dr
op%=TRUE:FX%=AX%:FY%=AY%
 1430 IF drop% THEN PROCdrop ELSE FOR W
=1TO10:NEXT
 1440 PRINT TAB(10,0);score%;SPC(5)
 1490 ENDPROC
 1499:
 1500 DEF PROChouse(X%,Y%,size%,colour%
)
 1510 GCOL0,colour%
 1520 MOVEX%,Y%:PLOT1,0,128
 1530 PLOT1,96,96:PLOT1,0,20
 1540 PLOT1,96-16*size%,0
 1550 PLOT1,0,16*size%-116
 1560 MOVE X%+96,Y%+224
 1570 PLOT1,96,-96:PLOT1,0,-128
 1580 ENDPROC
 1599:
 1600 DEF PROCmove
 1610 AX%=AX%-1
 1620 PRINT TAB(AX%,AY%);CHR$224;CHR$22
5;CHR$32
 1630 IF NOT drop% THENPX%=AX%*64:PY%=1
024-(AY%*32):VDU5:GCOL0,parcel-225:MOVE
PX%+32,PY%-4:VDUparcel:VDU4
 1640 IF AX%=1 THEN AX%=19:PRINTTAB(1,A
Y%);CHR$32;CHR$32:AY%=RND(10)
 1690 ENDPROC
 1699:
 1700 DEF PROCdrop
 1710 FY%=FY%+1
 1720 COLOUR parcel-225
 1730 PRINTTAB(FX%,FY%);CHR$parcel
 1740 PRINTTAB(FX%,FY%-1);CHR$32
 1750 COLOUR 7
 1760 IF FY%>19 THEN PROCtest:parcel=RN
D(3)+225:drop%=FALSE:PRINTTAB(FX%,FY%);
CHR$32
 1790 ENDPROC
 1799:
 1800 DEF PROCtest
 1810 ON parcel-225 GOTO 1840,1830,1820
 1820 IF FX%=15 score%=score%+50:PROCno
ise:ENDPROC
 1830 IF FX%=9 score%=score%+25:PROCnoi
se:ENDPROC
 1840 IF FX%=3 score%=score%+10:PROCnoi
se:ENDPROC
 1850 score%=score%-5:miss%=miss%+1
 1860 IF score%<0 THEN score%=0
 1870 IF miss%>20 THEN over%=TRUE
 1890 ENDPROC
 1899:
 1900 DEF PROCnoise
 1910 A%=100+25*(FX%DIV6)
 1920 SOUND 1,1,A%,5
 1990 ENDPROC
 1999:
 2000 DEF PROCend
 2010 CLS:COLOUR 1
 2020 *FX15,1
 2030 PRINT TAB(3,4);"Game Over"
 2040 PRINT TAB(2,8);"Your Score:"
 2050 COLOUR 2
 2060 PRINT TAB(14,8);score%
 2070 VDU23,1,1;0;0;0;
 2080 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

JOINING TWO OR MORE PROGRAMS

If you have two programs you want to join, but you don't want to type either of them back in since they're both long, then the easiest way to join them is by loading in the first program and typing:

*SPOOL"name"<return>

Then set the recorder to 'Record' and type

LIST<return>

Make sure you have a tape ready and the computer will then list the program out to tape as a file called "name". When the listing is complete, do not switch off the cassette yet but type:

*SPOOL<return)

Rewind the tape to the start of the file. You should now load your second program, and renumber it if any of the line numbers are the same as the SPOOLed program. Finally, type:

*EXEC"name"<return)

and your previous program will be read in from cassette, to join the program already residing in memory. You should ignore the error messages resulting from this last command.

RESCUING A BAD PROGRAM

by Colin Opie

One of the most dreaded error messages that your Electron will produce is 'Bad Program'. Once that has been displayed you will find that you can no longer access your current Basic program. ELBUG comes to your rescue with a program that will restore your program to health with almost magical efficiency. It can also be used to recover programs that refuse to load correctly from cassette.

The error message 'Bad program' is produced by Basic when it detects that the program in memory has been corrupted, or has been loaded incorrectly. Once the message has been issued, the user cannot directly modify this program in any way. LIST does not work and typing new lines has no effect. This can be disastrous if a small programming error results in a corrupted program for which you have no backup copy on tape. The RESCUE program presented here will search through a 'Bad Program' for the corrupted parts, and fix these in such a way that the program can once again be listed and edited. This may not completely restore the original program but it does allow you to gain access to the program so that damaged lines can be deleted or replaced.

USING RESCUE TO RECOVER BAD PROGRAMS

To use the RESCUE program, you must first type it in and save it on cassette (SAVE"RESCUE"). It is reasonably short but must be typed in very accurately. Once it is safely on tape, you can use it to recover a program.

When you get a 'Bad Program' message type the following:

```
PAGE=&5B00      <return>
CHAIN"RESCUE"   <return>
```

Changing the value of PAGE ensures that the RESCUE program occupies a separate area of memory to the corrupt program. The RESCUE program will then attempt to patch back together the program currently resident in memory. When it finishes, you should be able to list and edit your repaired program as normal.

```
10 REM Program  Rescue
20 REM Author   Colin Opie
30 REM Version  E0.1
40 REM ELBUG    December 1983
50 REM Program  subject to Copyright
60 :
70 ON ERROR GOTO 330
80 PROCinit
 90 REPEAT
100 PROCrecover
110 UNTIL finished
120 PAGE=P%
130 END
140 :
150 DEF PROCrecover
160 ?line=&0D: lenpos=line+3: count=1
170 IF ?line=&0D AND line?1=&FF THEN
finished=TRUE:ENDPROC
180 PRINT (line?1)*256+(line?2);
190 REPEAT
200 IF line?count<>&0D THEN count=cou
nt+1
210 IF line?count<>&0D AND line?count
<32 AND count>4 THEN line?count=35
220 IF count>250 THEN line?(count+1)=
&0D
230 UNTIL line?count=&0D
240 PRINT ~line
250 line=line+count: ?lenpos=count
260 ENDPROC
270 :
280 DEF PROCinit
290 P%=&E00
300 line=P%: line?1=0: finished=FALSE
310 ENDPROC
320 :
330 ON ERROR OFF
340 MODE 6:IF ERR=17 END
350 REPORT:PRINT" at line ";ERL
360 END
```

This program was adapted from a version for the BBC micro first published in BEEBUG Vol.1 No.8.

Explaining PAGE

Part of your computer's memory is reserved for its own use. The point where your own program starts is always stored for reference in a special variable called PAGE. On the Electron, this is normally set to &E00 (3584 in decimal), when you first switch the computer on. To check, type PRINT PAGE <return>. PAGE can be set to other values for special purposes, usually so that a program can be located in a different part of memory. It is possible, in this way, to have more than one program in memory at the same time. This happens when the RESCUE program and your program are in memory together.

Your first action on recovering a corrupt program should be to save it (as it is) on tape with SAVE"name", as this will safeguard against any further corruption of the program. Having saved the program, you should list it out and examine the lines of code. Some lines may contain rubbish, whilst others may have disappeared completely. Re-type or replace lines as needed.

Something to look out for are '#' characters appearing in the program after initial recovery. The rescue operation replaces any control characters in the program with the hash character (#), so that they can be easily recognised. Control characters cannot be allowed to remain, as they are likely to cause problems when LIST is attempted, and they should not be there anyway.

RECOVERING CASSETTE PROGRAMS

Sometimes, you will find it impossible to load correctly, a program from cassette and you will be plagued with a variety of error messages often followed by the words, "Rewind Tape". Trying to reload, will sometimes work but not always. Here we describe how the RESCUE program can once again, avert potential disaster.

Before attempting to reload the offending program from cassette, type the following line into your Electron:

*OPT2,0 <return>

Now proceed to load the bad program from cassette as before. This time, although error messages may still appear, the program will continue to be loaded into the computer, including any corrupted sections. Once the corrupt program is in memory, you can follow the steps already described, using the RESCUE program, to restore as much of your program as possible. You should also, at this stage, restore cassette loading to its normal state by typing:

*OPT <return>

Of course, this process will generally work only with your own programs, as you will need to know how to make corrections yourself, once RESCUE has done its job. It will not generally help with programs that you have bought or obtained elsewhere, especially if the Bad Program message arises as a result of copy protection put on the software by the manufacturer.

SOME FINAL HINTS

Normally, only a small part of any program is corrupted, and using RESCUE will completely restore the original program. Occasionally, even after using RESCUE, the recovered program will still contain a number of introduced errors, mostly in the form of unexpected or changed characters. The RESCUE program cannot help you with these faults, so you must find and repair them yourself - without the RESCUE program, even this would not be possible.

To avoid problems of this kind, you should save your programs frequently onto cassette. Saving two copies will also help to avoid loading problems later. Moving back to a previous version is usually much easier than recovering a damaged program and should be quicker overall.

FOOTBALL KRAZY

by Alan Dickinson

We present here, for your entertainment, the best game of football in town! In each game you get 90 seconds of fast moving, high scoring action, where you pit your wits against the computer's.

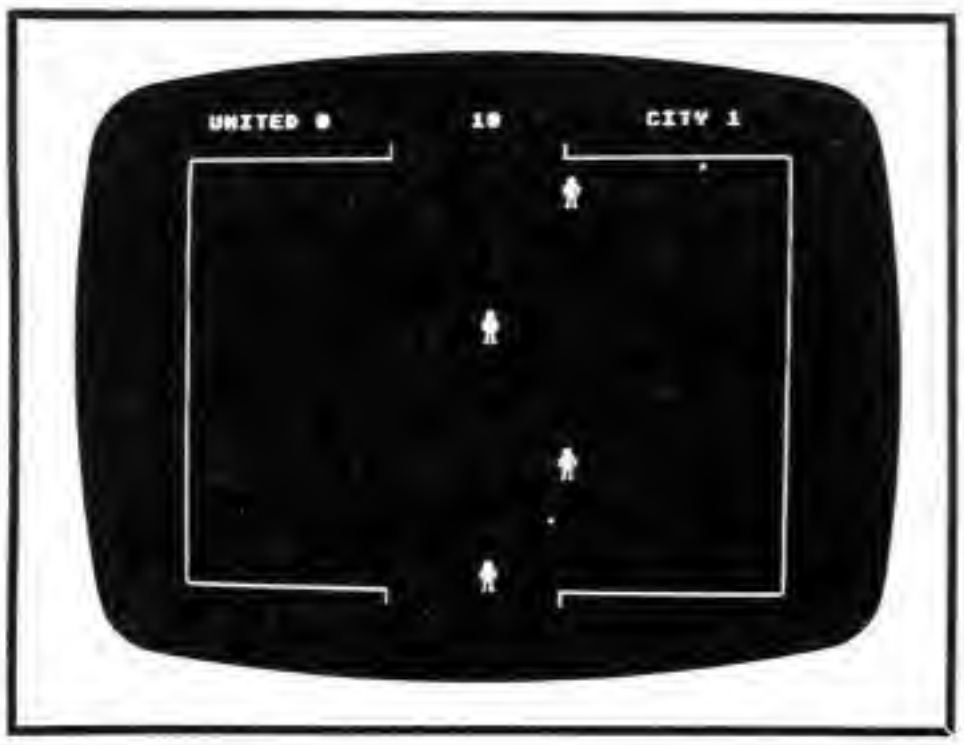
This short program, although written in Basic, gives a smooth, fast game of football, due to the use of good structuring techniques. This also allows you to modify easily the program, and change, for example, the playing time and the names of the teams.

In the game presented here, you are designated as the 'United' team with chequered shirts, playing against the computer, which controls the 'City' team in the striped shirts.

When the program runs, it initially asks you to enter the position in which you want your man to play (the goalie always stays on the bottom row). If you enter a value outside of the range asked for, then your man will be assigned to position 7. Position 26 is in the row next to your goalie, and 7 is in the row next to the computer's goalie. You are then asked for a value for the computer's defence tactic; this governs how the computer will respond in the game. The greater the number entered, the harder the game played. Try, for example, entering 26 in response to the position request, and 100 to the defence tactic. In this arrangement, you will find it virtually impossible to beat the computer as it will dribble, tackle, pass and score with unnerving accuracy, and run rings around your players. (Nobody in the office won, or even scored in this mode!)

To move your players, use the 'Z' key to go to the left, 'X' to move right while at any time, pressing '/' will allow an oncoming ball to pass through your player. A word of warning though, when a goal is scored, the computer moves its men back to the centre of the pitch, but leaves your men where they were when the goal was scored. You should, therefore, move your men back quickly, or the computer will instantly score another goal!

If you do wish to change the playing time, alter the value of Z% in line 400, and if you'd prefer different team names, then they should be changed in lines 610, 620, 1040, 1050, 1180 and 1190. (Don't forget to alter the relevant TAB values to take into account any change in length of the names you use.)




```
  10 REM Program: KRAZY
  20 REM Version: v4
  30 REM Author A.DICKINSON
  40 REM ELBUG December 1983
  50 REM Program subject to Copyright
  60 :
  70 ON ERROR IF ERR<>17 THEN ON ERROR
 OFF:MODE 6:REPORT:PRINT" at line ";ERL
:END
  80 MODE 4
  90 VDU23,1,0;0;0;0;0
 100 REPEAT
 110 PROCinit
 120 PROCtitle
 130 PROCstart
 140 PROCtune
 150 PROCplay
```

```
  160 PROCfinalscore
  170 REPEAT
  180 A$=GET$
  190 UNTIL A$="Y" OR A$="N"
  200 UNTIL A$="N"
  210 *FX15,1
  220 MODE 6
  230 END
  240 :
  250 DEF PROCinit
  260 VDU19,1,6,0,0,0
  270 VDU23,224,0,0,0,24,24,0,0,0
  280 VDU23,225,0,24,60,60,24,126,173,1
81
  290 VDU23,226,173,181,60,36,36,36,36,
102
  300 VDU23,227,0,24,60,60,24,126,129,1
89
  310 VDU23,228,129,189,60,36,36,36,36,
102
  320 VDU23,229,0,0,0,0,0,0,0,255
  330 VDU23,230,255,0,0,0,0,0,0,0
  340 VDU23,231,1,1,1,1,1,1,1,1
  350 VDU23,232,128,128,128,128,128,128
,128,128
  360 VDU23,233,255,1,1,1,1,1,1,1
  370 VDU23,234,255,128,128,128,128,128
,128,128
  380 VDU23,235,1,1,1,1,1,1,1,255
  390 VDU23,236,128,128,128,128,128,128
,128,255
  400 Z%=9000:REM GAME LENGTH
  410 B$=CHR$17+CHR$7+CHR$224
  420 P$=CHR$32+CHR$226+CHR$32+CHR$11+C
HR$8+CHR$8+CHR$8+CHR$32+CHR$225+CHR$32
  430 Q$=CHR$32+CHR$228+CHR$32+CHR$11+C
HR$8+CHR$8+CHR$8+CHR$32+CHR$227+CHR$32
  440 ENDPROC
  450 :
  460 DEF PROCtitle
  470 CLS:PRINTTAB(7,2)"FOOTBALL KRAZY"
  480 PRINTTAB(2,26)"Z = LEFT";
  490 PRINTTAB(2,28)"X = RIGHT";
  500 PRINTTAB(2,30)"/ = THROUGH-BALL";
  510 PRINTTAB(4,4)"Attack position 26-
7";
  520 INPUT A%
  530 IF A%<7 OR A%>26 A%=13
  540 PRINTTAB(4,6)"Defence tactic ";
  550 INPUT T%
  560 ENDPROC
  570 :
  580 DEF PROCstart
  590 S$=STRING$(3,CHR$32)
  600 CLS:SP%=0:SQ%=0
  610 PRINTTAB(1,1)"UNITED 0";
  620 PRINTTAB(28,1)"CITY   0";
  630 PRINTTAB(1,3)STRING$(38,CHR$229);
  640 PRINTTAB(1,30)STRING$(38,CHR$230);
  650 PRINTTAB(14,3)SPC11
  660 PRINTTAB(14,30)SPC11
  670 PRINTTAB(13,3)CHR$235;TAB(25,3)CH
R$236;TAB(13,30)CHR$233;TAB(25,30)CHR$2
34
  680 FOR Y%=4 TO 29
  690 PRINTTAB(0,Y%)CHR$231
  700 PRINTTAB(39,Y%)CHR$232
  710 NEXT
  720 P%=18
  730 PRINTTAB(P%,A%)P$;TAB(P%,28)P$;
  740 Q%=18:B%=18+RND(4)
  750 PRINTTAB(Q%,5)Q$;TAB(Q%,B%)Q$;
  760 ENDPROC
  770 :
  780 DEF PROCtune
  790 RESTORE
  800 REPEAT
  810 READ V%,D%:SOUND 1,-15*V%,72,D%
  820 UNTIL D%=0
  830 DATA 1,3,0,3,1,3,0,3
  840 DATA 1,2,0,1,1,2,0,1,1,4,0,2
  850 DATA 1,2,0,1,1,2,0,1,1,2,0,1,1,4,
0,2
  860 DATA 1,2,0,1,1,4,0,0
  870 ENDPROC
  880 :
  890 DEF PROCplay
  900 TIME=0
  910 REPEAT
  920 X%=20:D%=1-RND(2):Y%=16
  930 IF RND(10)>5 E%=1 ELSE E%=-1
  940 PRINTTAB(Q%,5)S$;TAB(Q%,B%)S$;TAB
(Q%,4)S$;TAB(Q%,B%-1)S$;
  950 Q%=18:PRINTTAB(X%,Y%)B$;
  960 PROCutd:PROCcity
  970 REPEAT
  980 PRINTTAB(19,1);TIME DIV 100;
  990 PROCutd:PROCcity:PROCball
 1000 UNTIL Y%<2 OR Y%>30 OR TIME>Z%+99
 1010 IF Y%<2 SP%=SP%+1
 1020 IF Y%>30 SQ%=SQ%+1
 1030 PRINTTAB(X%,Y%)" ";
 1040 PRINTTAB(1,1)"UNITED ";SP%
 1050 PRINTTAB(28,1)"CITY   ";SQ%;
 1060 FOR I%=0 TO 100 STEP 7
 1070 IF Y%<2 SOUND 1,-15,I%,2 ELSE SOU
ND 1,-15,110-I%,2
 1080 NEXT
 1090 UNTIL TIME>Z%+99
 1100 ENDPROC
 1110 :
 1120 DEF PROCfinalscore
 1130 PRINTTAB(15,16)" FULL TIME ";
 1140 PROCtune
 1150 REPEAT UNTIL TIME>Z%+500
 1160 CLS
 1170 PRINTTAB(2,4)"Result:"
 1180 PRINTTAB(4,6)"United ";SP%;
 1190 PRINTTAB(4,8)"City   ";SQ%;
 1200 IF SP%>SQ% PRINTTAB(4,11)"UNITED
WIN THE MATCH";
```

```
 1210 IF SP%=SQ% PRINTTAB(4,11)"IT'S A
DRAW ! ! !";
 1220 IF SP%<SQ% PRINTTAB(4,11)"CITY WI
N THE MATCH !!!";
 1230 PRINTTAB(2,14)"Again ?";
 1240 *FX15,1
 1250 ENDPROC
 1260 :
 1270 DEF PROCutd
 1280 IF INKEY(-98) AND P%>1 P%=P%-1:PR
INTTAB(P%,A%)P$;SPC1;TAB(P%,28)P$;SPC1;
:ENDPROC
 1290 IF INKEY(-67) AND P%<36 PRINTTAB(
P%,A%)SPC1;P$;TAB(P%,28)SPC1;P$;:P%=P%+
1:ENDPROC
 1300 PRINTTAB(P%,A%)P$;TAB(P%,28)P$;
 1310 ENDPROC
 1320 :
 1330 DEF PROCcity
 1340 IF X%<Q%+1 AND Q%>3 AND Y%<13+T%
Q%=Q%-1:PRINTTAB(Q%,5)Q$;SPC1;TAB(Q%,B%
)Q$;SPC1;:ENDPROC
 1350 IF X%>Q%+1 AND Q%<34 AND Y%<13+T%
 PRINTTAB(Q%,5)SPC1;Q$;TAB(Q%,B%)SPC1;Q
$;:Q%=Q%+1:ENDPROC
 1360 PRINTTAB(Q%,5)Q$;TAB(Q%,B%)Q$;
 1370 ENDPROC
 1380 :
 1390 DEF PROCball
 1400 N%=X%+D%
 1410 M%=Y%+E%
 1420 IF N%<1 OR N%>38 N%=X%:D%=-D%:SOU
ND 1,-15,100,1
 1430 IF M%=30 OR M%=3 IF N%<14 OR N%>2
3 M%=Y%:E%=-E%:SOUND 1,-15,120,1
 1440 IF M%=5 OR M%=B% PROCQman
 1450 IF M%=A% OR M%=28 PROCPman
 1460 PRINTTAB(X%,Y%)SPC1;TAB(N%,M%)B$;
 1470 X%=N%:Y%=M%
 1480 ENDPROC
 1490 :
 1500 DEF PROCPman
 1510 IF N%<P% OR N%>P%+2 ENDPROC
 1520 SOUND 1,-15,60,1
 1530 IF INKEY(-67) D%=-1
 1540 IF INKEY(-98) D%=1
 1550 IF NOT INKEY(-105) E%=-E%
 1560 ENDPROC
 1570 :
 1580 DEF PROCQman
 1590 IF N%<Q% OR N%>Q%+2 ENDPROC
 1600 SOUND 1,-15,52,1
 1610 IF RND(10+T%)>3 E%=1 ELSE E%=-E%
 1620 D%=0
 1630 R%=RND(10)
 1640 IF R%>7 D%=-1
 1650 IF R%<4 D%=1
 1660 IF T%<3 OR R%<4 ENDPROC
 1670 IF X%>25 D%=-1
 1680 IF X%<15 D%=1
 1690 ENDPROC
```

BEEBUGSOFT

Snake

Snake is a very addictive, fast moving arcade game. The purpose of the game is to direct the snake around the screen, eating up the randomly placed objects. With every bite that it takes, the tail grows longer and the snake moves faster. Just to complicate things, there are also certain scattered obstacles which must be avoided.

To even complete the first screen requires skill and concentration. Further frames use different shaped playing areas, which as well as adding variety, require much greater skill to complete.

This game is much more exciting than many games of a similar nature, and once played, proves to be surprisingly compulsive.

Snake costs £4.00 inc VAT. Please add 50p post & packing.
Overseas orders: £5.50 inc post & packing - VAT not charged.
Send order with membership number to:
BEEBUGSOFT, P.O. Box 109, High Wycombe, Bucks HP11 2TD.

IF YOU WRITE TO US

BACK ISSUES (Members only)

All back issues will be kept in print (from November 1983). Send 90p per issue PLUS an A5 SAE to the subscriptions address. Back copies of BEEBUG are available to ORBIT members at this same price. This offer is for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the advertising supplements are not supplied with back issues.

Subscription and Software Address

ELBUG
PO BOX 109
High Wycombe
Bucks
HP11 2TD

SUBSCRIPTIONS

Send all applications for membership, and subscription queries to the subscriptions address.

MEMBERSHIP COSTS:
£5.90 for 6 months (5 issues)
£9.90 for 1 year (10 issues)
European Membership £16 for 1 year.
Elsewhere (Postal zones)
Zone A £19, Zone B £21, Zone C £23

SOFTWARE (Members only)

This is available from the software address.

IDEAS, HINTS & TIPS, PROGRAMS AND LONGER ARTICLES

Substantial articles are particularly welcome and we will pay around £25 per page for these, but in this case please give us warning of anything that you intend to write.

We will also pay £10 for the best Hint or Tip that we publish, and £5 to the next best. Please send all editorial material to the editorial address opposite. If you require a reply it is essential to quote your membership number and enclose an SAE.

Editorial Address

ELBUG
PO Box 50
St Albans
Herts
AL1 2AR

ELBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams. Technical Editor: Philip Le Grand.
Production Editor: Phyllida Vanstone. Technical Assistant: Alan Webster.
Managing Editor: Lee Calcraft.
Thanks are due to David Graham, Sheridan Williams, Adrian Calcraft and Matthew Rapier for assistance with this issue.

Printed in England by Staples Printers St Albans Limited at The Priory Press.